THESIS

GLOBAL COMBAT SUPPORT SYSTEM-MARINE CORPS PROOF-OF-CONCEPT FOR DASHBOARD ANALYTICS

by

Timothy J. Leonard
Philip Gallo

December 2014

Thesis Co-Advisor:	Bryan Hudgens
Thesis Co-Advisor:	Anthony Kendall

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE GLOBAL COMBAT SUPPORT SYSTEM-MARINE CORPS PROOF-OF-CONCEPT FOR DASHBOARD ANALYTICS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Timothy J. Leonard and Philip Gallo			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Installation and Logistics, HQMC			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Global Combat Support System – Marine Corps (GCSS-MC) was created to combine both Logistics and Supply capabilities into one system. The existing systems of Asset Tracking Logistics and Supply System (Atlass) and PC Marine Corps Integrated Maintenance Management System (MIMMS) required external interfaces to merge data. This new system was established to streamline the way the Marine Corps allowed commanders to make logistic and supply decisions. Unfortunately, GCSS-MC was not implemented with any Logistics or Supply Analytics, which required users to manipulate Microsoft Office products such as Access and Excel to analyze data. The objective of this thesis was to design, develop, and test a proof-of-concept prototype utilizing Oracle WebCenter to create dashboard analytics that allow commanders at all levels to make informed near-real time decisions. This was achieved by using the following: a modern design approach; the Model-View-Controller; a state-of-the-art development pattern; Oracle Application Development Framework; and powerful development tools such as Oracle JDeveloper and Oracle WebCenter Portal Builder.

14. SUBJECT TERMS GCSS-MC, Oracle ADF, USMC, Readiness			**15. NUMBER OF PAGES** 185
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

GLOBAL COMBAT SUPPORT SYSTEM-MARINE CORPS PROOF-OF-CONCEPT FOR DASHBOARD ANALYTICS

Timothy J. Leonard
Major, United States Marine Corps
B.S., Embry Riddle Aeronautical, 2002

Philip Gallo
Captain, United States Marine Corps
B.A., American Military University, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF BUSINESS ADMINISTRATION

from the

NAVAL POSTGRADUATE SCHOOL
December 2014

Authors:	Timothy J. Leonard Philip Gallo
Approved by:	Bryan Hudgens Thesis Co-Advisor
	Anthony Kendall Thesis Co-Advisor
	William Gates Dean, Graduate School of Business

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Global Combat Support System – Marine Corps (GCSS-MC) was created to combine both Logistics and Supply capabilities into one system. The existing systems of Asset Tracking Logistics and Supply System (Atlass) and PC Marine Corps Integrated Maintenance Management System (MIMMS) required external interfaces to merge data. This new system was established to streamline the way the Marine Corps allowed commanders to make logistic and supply decisions. Unfortunately, GCSS-MC was not implemented with any Logistics or Supply Analytics, which required users to manipulate Microsoft Office products such as Access and Excel to analyze data. The objective of this thesis was to design, develop, and test a proof-of-concept prototype utilizing Oracle WebCenter to create dashboard analytics that allow commanders at all levels to make informed near-real time decisions. This was achieved by using the following: a modern design approach; the Model-View-Controller; a state-of-the-art development pattern; Oracle Application Development Framework; and powerful development tools such as Oracle JDeveloper and Oracle WebCenter Portal Builder.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AAC	Activity Address Code
ADM	Acquisition Decision Memorandum
AIT	Automated Identification Technology
ATLASS1	Asset Tracking, Logistics, and Supply System 1
COTS	Commercial Off the Shelf
DAA	Designated Approval Authority
DODAAD	DOD Activity Address Directory
DOORS	Dynamic Object-Oriented Requirements System
ERP	Enterprise Resource Planning
FDR	Functional Design Review
GCSS-MC	Global Combat Support System – Marine Corps
IPR	Interim Program/Progress Review
ITEM APPS	Item Applications
ITV	In Transit Visibility
LCM	Logistics Chain Management
LMIS	Logistics Management Information System
MAGTF	Marine Air/Ground Task Force
MCCDC	Marine Corps Combat Development Command
MCHS	MC Hardware Suite
MCSC	Marine Corps Systems Command
MCTFS	Marine Corps Total Force System
MDR	MERIT Data Repository
MDSS II	MAGTF Deployment Support System II
MEF	Marine Expeditionary Force
MERIT	Marine Corps Equipment Readiness Information Tool
MIMMS	Marine Corps Integrated Maintenance Management System
NMCI	Navy/Marine Corps Internet
PMO	Project Management Office

SABRS	Standard Accounting, Budgeting and Reporting System
SASSY	Supported Activity Supply System
TO/E	Table of Equipment
T/MR	Table of Manpower Requirements
TFSMS	Total Force Structure Management System
VO	View Object

ACKNOWLEDGMENTS

We would like to thank our families for their sacrifice, patience, and support through this entire experience. We would also like to thank our advisors, Bryan Hudgens and Walter Kendall for their wisdom, guidance and continued support, both in the classroom and in completing this project. Finally, we would like to thank the interns whom provided the technical expertise utilizing Oracle products and the GCSS-MC Program Office for their time and support invested in our research. Thank you for all those that have supported us through our time at the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

In 2004, the Office of the Secretary of Defense (OSD) designated GCSS-MC as an Acquisition Category (ACAT) 1AM special interest program. An ACAT 1AM Program is a Major Defense Acquisition Program; according to the Defense Acquisition University,

> an MDAP is a program that is not a highly sensitive classified program and that is designated by the Under Secretary of Defense for Acquisition, Technology and Logistics (USD(AT&L)) as a MDAP; or that is estimated to require eventual expenditure for research, development, test, and evaluation (RDT&E), including all planned increments, of more than $480 million (Fiscal Year (FY) 2014 constant dollars) or procurement, including all planned increments, of more than $2.79 billion (FY 2014 constant dollars). [1]

GCSS-MC is the primary technology enabler for the Marine Corps Logistics modernization strategy. It will replace a portfolio of 200 fragmented legacy logistics systems that provide the backbone for all logistics information required by the Marine Air Ground Task Force (MAGTF). The core is modern, commercial-off-the-shelf enterprise resource planning (ERP) software (Oracle 11i e-Business Suite). GCSS-MCs design is focused on enabling the warfighter to operate while deployed with reach back from the battlefield. The systems previously in use were created to support specific functional areas within the Supply and Logistics fields. Most importantly, they were not designed to work in austere environments.

Lessons learned from Operations Desert Storm and Iraqi Freedom revealed that deployable, integrated technology is more than a “nice-to-have” for the warfighter. Systems that could not communicate with each other resulted in huge order-fulfillment lag times, redundant ordering, choked pipelines, and warfighter uncertainty. Commanders compensated for lack of information by stockpiling supplies to cover any eventuality and, as a result, sacrificed some of the agility and flexibility they needed to fight on a non-traditional battlefield. Hauling huge amounts of inventory around equates to a reduction in combat power [2].

The near-real time information GCSS-MC provides will give Marines vastly improved asset visibility, reduced customer wait time, and a lighter, more mobile fighting force. Timely, accurate data also enables faster, better decision making, a prerequisite for improved combat effectiveness of the MAGTF. GCSS-MC can go wherever Marines go and provide consistent, accurate information up and down the supply chain [2].

In 2010, milestone C was completed and the initial fielding of GCSS-MC to III Marine Expeditionary Force (MEF) had begun. During Milestone C, recommendations are made and approvals to enter the Production and Development phase are sought. The initialing field of Block 1 was completed in December 2012, and all units have been cutover to GCSS-MC. GCSS-MC Increment 1 provides five functional logistics capabilities of Request Management, Supply, Maintenance, Financial, and System Administration. This is further decomposed to address those supported logistics capabilities/business process capabilities necessary to achieve the requirements outlined in the GCSS-MC/Logistics Chain Management (LCM) Increment 1 (Block 1), Capability Production Document (CPO). This includes Request Management, Order Management, Inventory Planning, Demand Planning, Maintenance Management, Inventory Capacity Operations, Warehouse Management, Distribution Management, Procurement Management, Asset Management, Task Organization, Customer Management, and Sourcing. Block 1 was the phasing in of GCSS-MC to replace all legacy supply and logistic systems.

B. OBJECTIVE

The objective of this thesis was to design, develop, and test a proof-of-concept prototype utilizing Oracle WebCenter to create dashboard analytics that allow commanders to make near-real time logistical decisions. These decisions would incorporate readiness throughout the hierarchy of Marine Corps units as well as equipment repair times for commonly deadlined equipment. This would allow commanders to employ systems that are required for assigned missions utilizing historical data generated through the dashboard.

C. PROBLEM STATEMENT

The timely and accurate reporting of unit supply and maintenance readiness is the cornerstone of a unit's operational capability. The current infrastructure of GCSS-MC does not give unit commanders the ability to quickly analyze their current readiness levels in real time. Users are exporting information into external software, typically Microsoft Excel and Microsoft Access, to generate analytics at the current time. Though these products have the ability to present information requested, transferring data between systems is time consuming. It is imperative that unit commanders have accurate information to make decisions that can result in mission success. Without knowing the true readiness of equipment, commanders will be unable to assign appropriate missions to appropriate commodities. This research will provide commanders some of the necessary tools to make these decisions without sorting through a multitude of reports as is the current status quo with GCSS-MC.

D. SCOPE AND METHODOLOGY

1. Scope

This research is focused on the design and development of a dashboard environment proof-of-concept web application to be utilized with GCSS-MC. This dashboard will be centered around six individual Use Cases that will allow commanders at all levels of operation control to make informed timely decisions on unit supply and equipment readiness. Installation and Logistics (I&L), Headquarters Marine Corps provided the historical data for this research.

2. Methodology

The methodology used for this research includes the following steps:

- Conduct a literature review and evaluation of GCSS-MC and Oracle WebCenter;
- Complete a requirements/gap analysis of current Marine Corps logistics systems;
- Learn the Oracle tools used for development of application framework;
- Utilize the Oracle Application Development Framework (ADF) developed by Oracle to create dash board environment;
- Create analytics from data in the GCSS-MC database;
- Develop a conceptual model;
- Build the prototype;
- Assess the prototype;
- Capture shortcomings required by users;
- Revise dashboard and analytics based on user feedback.

3. Primary Research Questions

Can web-based application analytics using Oracle ADF and WebCenter technology reduce the gap between the current limitations of GCSS-MC analytic capabilities and the actual analytics needed by the GCSS-MC users? How well can this application use and access existing logistics databases?

4. Benefits of Study

The current framework of GCSS does not display requested information quickly and requires users to have detailed knowledge of information required to pull reports. The successful implementation of a dashboard environment in GCSS-MC will allow commanders to make real time decisions without the need to pull and analyze required reports through Oracle Discovery Reports Users module within GCSS-MC.

E. ORGANIZATION OF THESIS

This thesis consists of six chapters:

- **Chapter I: Introduction**. This chapter gives a general outline of the problem with a description of research scope and methodology, and the organization of the thesis.
- **Chapter II: GCSS - MC**. This chapter discusses the background of GCSS-MC.
- **Chapter III: System Requirements' Analysis**. This chapter discusses the application requirements' analysis and defines the data model and Use Cases.
- **Chapter IV: Development Method and Architecture**. This chapter discusses the development approach used in this research, the Oracle Application Development Framework, and the tools used for development, including JDeveloper, SQL Developer, and WebCenter.
- **Chapter V: Application Development**. This chapter describes the elements of the system as well as the application features and dashboard available to the users.
- **Chapter VI: Conclusion**. This chapter summarizes the key findings and conclusions drawn from this thesis and offers recommendations for future research areas.

THIS PAGE INTENTIONALLY LEFT BLANK

II. GLOBAL COMBAT SUPPORT SYSTEM – MARINE CORPS

A. GLOBAL COMBAT SUPPORT SYSTEM – MARINE CORPS

In December 2012, GCSS-MC became the system of record for the Marine Corps Supply and Logistics community. The implementation of this program replaces an approximately 30-year-old collection of programs used within these communities. The previous programs of record, Asset Tracking Logistics and Supply System (ATLASS)/Supported Activities Supply System (SASSY) and Marine Corps Integrated Maintenance Management System (MIMMS), which were the primary means of managing maintenance and supply readiness, have since been retired. The inability of these two programs to give real time information was a large factor in the decision to implement this new software. GCSS-MC is the technology centerpiece of the Logistics Modernization (LogMod) Program. LogMod focuses on integrating a new organizational structure with streamlined processes and modern technology. GCSS-MC will use Oracle's E-Business Suite software for its application.

Only registered users who have an assigned Billet Identification Code (BIC) for their unit can access. Once an individual is registered to access GCSS-MC, the unit's Using Unit Account Manager (UUAM) grants specific roles and responsibilities within the system. The UUAM assigns all roles and responsibilities for the expected tour length of the individual assigned.

Once an individual is granted access, roles, and responsibilities, he will be able to access the user's GCSS-MC dashboard, which is customizable to show the user's most often utilized links. This will allow for easier navigation of the system. Figure 1 shows a generic user dashboard upon initial log on to GCSS-MC.

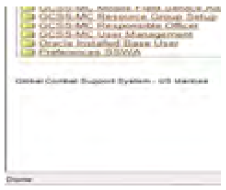

GCSS-MC Resource Group Setup
GCSS-MC Responsible Officer
GCSS-MC User Management
Oracle Installed Base User
Preferences SSWA
Global Combat Support System - US Marines
Done

Figure 1. GCSS-MC Login Dashboard.

B. HOW DOES GCSS-MC WORK?

Oracle designed GCSS-MC using Commercial-Off-The-Shelf (COTS) software and customizing it to meet the needs of the Marine Corps. GCSS-MC is comprised of several relational databases that are managed by Space and Naval Warfare Systems Command (SPARWAR). GCSS-MC is based on the implementation of Oracle e-Business Suite 11i as the core software package. This is the same infrastructure that Oracle WebCenter utilizes that allows for ease of interface between the two programs. A database is used to store information that allows for simple retrieval of stored information. Databases are composed of tables with rows and columns of information. Data in a table can be related according to primary keys or concepts, and the ability to retrieve related data from a table is the basis for the term relational database.

III. SYSTEM REQUIREMENTS' ANALYSIS

A. XXMC R001 DATABASE

The XXMC R001 database, provided by the GCSS-MC program office, contained the required information to obtain the analytics for the following Use Cases. The XXMC R001 database is a daily extract from multiple existing databases within GCSS-MC. Appendix A lists the relationships within the XXMC R001 Database.

B. USE CASES

The primary purpose of this research topic is to provide business analytics to unit commanders for the following six Use Cases. Each Use Case brings its own decision making points to the forefront for commanders to act on to ensure that supply and maintenance readiness is maintained for operational readiness. Use Cases facilitated the development of this application. The Use Cases are important because they define the parameters of data, which is required to pull analytics from the existing XXMC R001 database. By analyzing Use Cases, we were able to understand what information commanders would want to capture in order to provide accurate decisions in the employment of their units. The following six Use Cases were presented and analyzed to provide the required analytics for this proof-of-concept:

- Unit Readiness Report: Displays unit maintenance readiness
- Maintenance Effort Report: Displays man-hours per Service Request
- Secondary Reparable Report: Displays man-hours and trends of Secondary Reparable repair times
- Cost Report: Displays financial data by unit for requisitions
- GCSS-MC Maintenance Production Report: Display all tasks associated to individual service requests

- Document Number Report: Displays both open and closed requisitions in one report

The business analytics that are being presented in this proof-of-concept are described next.

1. Readiness Report (Alpha)

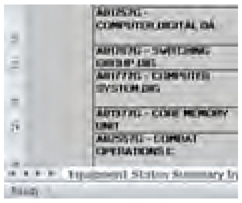
This report allows the viewer to determine aggregated readiness over time, specific equipment, unit, Major Subordinate Command (MSC), and Marine Expeditionary Force (MEF). Unit readiness is the ability to provide capabilities required by the combatant commanders to execute their assigned missions. This is derived from the ability of each unit to deliver the outputs for which it was designed. Unit readiness is reported by the military service [3]. Units report the status for both equipment and supplies on hand (S-Rating) and equipment condition (R-Rating). The S-rating is the material measurement of an organization's possessed equipment quantity against its designed requirement. The R-rating indicates the material condition of the organization's possessed equipment [3]. The equipment selected for reporting will either be Mission Essential Equipment (MEE) or principal end item (PEI). Marine Corps Bulletin 3000, which is published annually, lists all MEE and PEI required for reporting purposes.

The Total Force Structure Management System (TFSMS) is the authoritative source for obtaining a unit's table of equipment (T/E) data for ground equipment. The T/E-prescribed wartime requirement appears in the AAO column, which will be changed in the future to read Unit T/E Requirement (UTR) [3]. The On-Hand column on the units Mechanized Allowance Listing (MAL) is the physical number of equipment on hand for each Table of Authorized Material Control Number (TAMCN) and National Stock Number (NSN).

The information required from GCSS-MC and TFSMS to create a readiness report includes the following:

- TAMCN: Table of Authorized Material Control Number is a supply-unique description code given to a Principal End Item consisting of (1) Commodity Designator (A – E), and (2) Item Number
- T/E Qty: Table of Equipment Quantity generated by TFSMS
- MAL RQMT: Mechanized Allowance List Requirement generated from TFSMS
- MAL AUTH: Mechanized Allowance List Authorized generated from TFSMS
- MAL PLND: Mechanized Allowance List Planned generated from TFSMS
- MAL UNFD: Mechanized Allowance List Unfunded generated from TFSMS
- MAL SPALOW: Mechanized Allowance List Special Allowance; approved from higher headquarters excess gear is authorized
- MAL CMDADJ: Mechanized Allowance List Command Adjusted; Allowance approved by higher headquarters that differs from T/E
- MAL OH: Mechanized Allowance List On hand quantity
- IB OH: Install base On hand quantity
- 01A OH: Perpetual inventory On hand quantity
- IB DL: Install Base deadlined equipment
- SR DL: Service request deadline equipment

Figure 2 is an example of an Equipment Status Report pulled from GCSS-MC Discovery Report User.


A01252G - COMPUTER,DIGITAL DA

A01767G - SWITCHING GROUP,DIG

A01772G - COMPUTER SYSTEM,DIG

A01977G - CORE MEMORY UNIT

A02557G - COMBAT OPERATIONS C

Figure 2. GCSS-MC Equipment Status Report.

Figure 2 shows the readiness of a unit by comparing the T/E On Hand against the MAL On Hand to get the Unit Supply Readiness. It also shows the Unit Equipment Readiness by comparing the MAL On Hand and the IB deadline and SR deadline columns. GCSS-MC cannot display this information readily so users generate it by running ad hoc reports in Discovery Reports Users, exporting to an excel format, and then loading the exported report into a Microsoft Access Database. These additional steps are time consuming and do not allow real time results. All applicable information is located on records within the GCSS-MC XXMC Tables that we will create the web-application to produce a dashboard that will generate readiness based on certain pre-designated criteria.

Figure 3 displays the table organization that is being utilized to prepare dashboard readiness display and the required fields within the R001 Database that were utilized to pull these analytics. The primary fields that are being utilized to search for the needed criteria are:

- Service Request: Unique identifier that defines a customer profile and the supplier of the appropriate goods and/or services and the item needed
- MARES: Marine Automated Readiness Evaluation System
- Operational Status: Code stating status the equipment is in i.e. (deadline, operational)

[redacted]

Figure 3. Tables for Readiness Dashboard.

2. Maintenance Effort Report (Bravo)

This report allows the viewer to determine how much effort was expended on the maintenance performed (HOURS). It will identify by type of service request (Maintenance, Preventive Maintenance Checks and Services (PMCS), Modifications (MOD), Cannibalization, Selective Interchange, and Calibrations (CAL)) the number of tasks and hours associated with the service request (Work Performed) and categorize the work and effort into different types of service requests and the tasks associated to them. Figure 4 displays the format within GCSS-MC where work hours are entered into service requests.

Figure 4. Work Hours Screenshot.

Currently within GCSS-MC users have to pull each service request individually and then add manually all hours for each task that was completed under the assigned requests. Service requests have a one-to-many relationship with tasks. The purpose of this report is to provide an overall picture of the amount of work hours performed for each task. Analysis will identify any correlation between tasks that take the longest to fix and the time an item is actually in the maintenance cycle. This report will also have the capability to identify units that have adopted maintenance practices and procedures that may be utilized to increase readiness among all units with the same echelon of maintenance abilities. Figure 5 illustrates the tables and relationships that were required to be pulled from the R001 Database to create the output for this Use Case. It also displays the format that this report will be displayed once it is created.

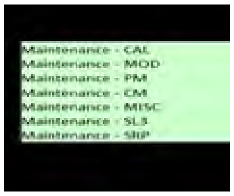

Maintenance - CAL
Maintenance - MOD
Maintenance - PM
Maintenance - CM
Maintenance - MISC
Maintenance - SL3
Maintenance - SRP

Figure 5. GCSS-MC Maintenance Effort Proposed.

3. Secondary Reparable Report (Charlie)

Allows the viewer to determine the effort associated to repair secondary reparable items. This report will identify the time associated to the repair, number of times it has been in the maintenance cycles (intervals) and aggregated by principle end item (TAMCN) or weapons system.

Secondary reparable items require more intense and time-consuming maintenance to bring equipment back to an operational status. This report will identify reoccurring issues with certain secondary reparable items that are entering the maintenance cycle continually. The two levels of maintenance (LOM) that perform work on secondary reparable are:

a. Field Level

Organizational Level (1st): This level sustains equipment in a mission capable condition status and is both preventative and corrective in nature. The responsibility of performing organizational maintenance is the assigned user's responsibility. This includes expeditious assessment and maintenance conducted under battlefield conditions. Tasks normally associated with organizational maintenance are inspecting, servicing, adjusting, testing, and replacing parts and components both major and minor assemblies [4].

Intermediate Level (2nd): The performance of intermediate level maintenance requires higher levels of technical training, specialized tools, and facilities performed by specially trained mechanics and technicians per individual training standards and technical publications [4]. This includes inspection/in-depth diagnosis, modification, replacement, adjustment, and limited repair or evacuation/disposal of principal end items and their selected reparables and components/subcomponents. Calibrations and repair of Test, Measurement and Diagnostic Equipment (TMDE), welding, and software upgrades are also performed at the intermediate level [4].

b. Depot Level

In this level, maintenance actions taken on material or software involve the inspection, repair, overhaul, or the modification or reclamation (as necessary) of weapons systems, equipment end items, parts, components, assemblies, and sub-assemblies that are beyond field maintenance capabilities, and/or are authorized and directed by DC I&L [4].

In this report, a reparable Item is "an item of supply subject to economical repair and for which the repair (at either depot or field level) is considered in satisfying computed requirements at any level" [3]. An NSN is "simply the official label applied to an item of supply that is repeatedly procured, stocked, stored, issued, and used throughout federal supply system" [5]. In an NSN, the specific digits translate as:

- First Four Digits: The Federal Supply Class
- Digits 5 and 6: Country of origin
- Digits 7 to 13: Sequentially assigned and unique to each NSN

Figure 6 illustrates the tables and relationships that were required to be pulled from the R001 Database to create the output for this Use Case. It also displays the format that this report will be displayed once it is created.

[redacted]

Figure 6. GCSS-MC Secondary Reparable Proposed.

4. Cost Report (Delta)

This report captures the service request cost associated to each service request by Unit, MSC, and MEF.

Budget execution and the tracking of authorized funds are essential for all units in the Marine Corps. GCSS-MC does not allow units to pull a consolidated listing of all service requests and cost associated. Fiscal personnel have the ability to pull a Budget Execution Report for their unit. The budget execution report will give a listing of all document numbers and the total cost associated to that document number. The way that it is provided, this report cannot be broken down to identify which document number belongs to a certain service request. Additionally, each time a user needs to pull this information requires logging in to the Discovery Reports View in GCSS-MC and entering the required information and date range for the period requested. Timely quarterly reconciliation for maintenance costs is difficult to complete since the required information is not readily available. It is difficult to properly brief unit commanders on costs of service request unless each individual service request is pulled and dollar values are totaled individually. The information required to identify the proper allocation of funds utilized by a unit to purchase parts are:

- OPBUD: Operating Budget
- MRI: Major Command Recipient Identifier
- WRI: Work Center Identifier
- BEA: Budget Execution Activity
- BESA: Budget Execution Sub-Activities
- COST JON: Job Order Number
- LOA: Line of Accounting

Figure 7 illustrates the tables and relationships that were required to be pulled from the R001 Database to create the output for this Use Case. It also displays the format that this report will be displayed in once it is created.

Figure 7. GCSS-MC Cost Report Proposed.

5. GMRT Maintenance Production Report (Echo)

The current GMRT Maintenance Production report does not show analysts the associated tasks for each defect identified within a Service Request or for pending requisitions awaiting approval. The GMRT Maintenance Production Report was designed to view maintenance related information (tasks and open parts requirements) on one single report, which allows analysts to better identify trends and pin-point procedural problems while assisting commanders and commodity managers with a more accurate assessment of equipment undergoing maintenance. This results in increased readiness reporting accuracy and shorter maintenance cycle times.

The information on the Maintenance Production Report (MPR) is processed and edited when Maintenance submits a Service Request. This report will give maintenance managers at all levels the visibility of active Service Requests in their shops. The purpose of this report is to provide the complete history of an item in the maintenance cycle. Supply status on this report will automatically update at the same time the DASF is updated. The Maintenance Production Report is used by the maintenance personnel in the same ways in which supply personnel uses the Due and Status File (DASF); it also contains much of the same information.

As seen previously, to validate all parts on order for an individual service request the user has to view each request individually and then export the information to an excel spreadsheet to analyze. The approval process for parts requisitions has multiple stages, and parts can be help-up by individuals if their work queue is not checked on a daily basis. This report would allow maintainers the ability to verify if parts have not been requisitioned in a timely manner by the approving official. Figure 8 illustrates the tables and relationships that were required to be pulled from the R001 Database to create the output for this Use Case. It also displays the format that this report will be displayed in once it is created.

[redacted]

Figure 8. GCSS-MC Maintenance Production Report Example.

6. Document Number Report (Foxtrot)

This report is designed to focus on all outstanding repair part requisitions for OPEN and CLOSED service requests. The report may be used to indicate trends in faulty parts, changes in repair procedures, whether debriefs to parts are being performed prior to closure of a service request, and supply problems. It also provides a general idea of the usage volume for particular parts by NIIN and whether or not these parts are seriously hindering the repair cycle. This report can be very beneficial in the reconciliation and validation process between the maintenance and supply commodities.

The Due and Status File (DASF) is a computerized record of all outstanding requisitions and stock replenishment requisitions within the command and their most recent status. Reconciliations are done on a bi-weekly schedule between the supply and the maintenance clerks utilizing both the MPR and the DASF. Figure 9 illustrates the report layout that is currently being utilized by units to view Due and Status file information. The image is unable to display both open and closed document numbers by service request.

[redacted]

Figure 9. GCSS-MC DASF Report.

Figure 10 illustrates the tables and relationships that were required to be pulled from the R001 Database to create the output for this Use Case. It also displays the format that this report will be displayed in once it is created.

[redacted]

Figure 10. GCSS-MC DASF Proposed Display.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DEVELOPMENT METHOD AND ARCHITECTURE

This chapter gives a description of the development methods used in the creation of our web-based application. It starts with a brief history of Oracle and Oracle fusion, which is the structure and encompasses all of the instruments we will be using. Next, we will describe our method of development. The Model-View-Controller and Oracle Application Development Framework (ADF) is the framework that we chose to build this proof-of-concept. We examine the four layers of Oracle ADF: Business layer, Model layer, Control layer, View layer. We also provide key benefits of Oracle ADF. We describe the main integrated development platform, Oracle JDeveloper and the Oracle WebLogic server. Then, we provide a description of the main components of Oracle WebCenter. To conclude, we explain both the modeling tool Oracle Structured Query Language (SQL) Development Data Modeler and the graphical management tool Oracle SQL Developer.

A. ORACLE

The corporation was founded by Larry Ellison, co-founded by Bob Miner and Ed Oates in 1977. This publicly traded company (NYSE: ORCL) is headquartered in Redwood City, California, and specializes in database software and hardware products [6]. Once only known for their database abilities, though the years, Oracle has expanded throughout the technology industry providing services ranging from servers, storage, financial, human resources (HR), enterprise resource planning (ERP), customer relationship management (CRM), supply chain management (SCM), and cloud computing. They have expanded through timely acquisitions. Since 2005, Oracle has grown very rapidly by acquiring 76 companies [6]. The acquisition of these companies brought cutting-edge technology and new products into Oracle. These new technologies coupled with the leadership at Oracle have propelled the company into an industry leader.

B. ORACLE FUSION

1. Oracle Fusion Architecture

Oracle Fusion Architecture (OFA) is the overall framework for building applications. This common architecture allows new applications to be built rapidly by reusing the common services. With this architecture, we can create applications from current services and not have to rely on coding every application [7]. As depicted in Figure 11, the Oracle Fusion Applications are built in the Oracle Fusion Middleware stack and utilize the Oracle Database [8].

Figure 11. Oracle Fusion Architecture, from [8].

2. Oracle Fusion Applications

Oracle Fusion Applications are 100 percent open standards-based which were designed from the ground up to meet their customers' global demands. The open standard foundation reduces risk for implementation and integration. These new applications were created to work with other enterprise and industry applications to support better business decision-making and to automate business procedures. Oracle has seven core product enterprise applications:

- Customer Relationship Management (CRM)
- Financial Management
- Governance, Risk, and Compliance (GRC)
- Human Capital Management (HCM)
- Procurement
- Project Portfolio Management (PPM)
- Supply Chain Management (SCM)

3. Oracle Fusion Middleware

Oracle Fusion Middleware is the software layer that connects the fusion applications. The Fusion Middleware is the infrastructure that facilitates creation of business applications and provides framework for service-oriented architecture (SOA) [9]. Oracle Fusion Middleware includes Web servers, application servers, content management systems, and support for development and deployment and management of applications. This collection of open standards-based software includes Java Enterprise Edition 5 compliant environment, developer tools, integration services, business intelligence, collaboration, and content management [9]. Figure 12 provides a summary of Oracle Fusion Middleware.

Figure 12. Oracle Fusion Middleware Solution Overview, from [9].

C. MODEL-VIEW-CONTROLLER

The basic tasks which a Fusion Application performs are: Data Access, Business logic implementation, User interface display, User interaction, and Application page flow. The Model-View-Controller groups the basic building blocks of a Fusion Application into three separate elements. This design pattern provides the architecture for graphical user interface (GUI). The three elements are: the model, the view, and the controller. Each of these elements performs a distinct role: [10]

- **Model:** The model represents data and the rules that govern access to and updates of this data and business logic implementation.

- **View:** It provides the user interface display to the application and raises events to the controller. It specifies exactly how the model data should be presented on the page.

- **Controller:** It manages application flow. The controller translates the user's interactions with the view into actions that the model will perform [Rob Eckstein].

Figure 13 illustrates the three elements.

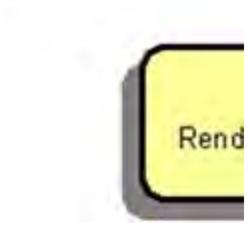


Figure 13. The Model-View-Controller elements and interactions, from [11].

D. ORACLE APPLICATION DEVELOPMENT FRAMEWORK (ADF)

1. Oracle ADF Overview

ADF is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards, and open-source technologies [12]. Java EE is a standard, robust, and secure platform that many of today’s enterprise applications utilize. Java EE, however, is very complex and requires significant coding requirements. Oracle ADF framework provides integrated infrastructure solutions and simplifies Java EE development by minimizing code writing. Oracle ADF accelerates development by adhering to Java EE standard patterns and practices allowing the developers to focus on the logic of application creation, which greatly reduces the amount of coding requirements. Oracle ADF is the framework and is directly supported by Java development tool Oracle

JDeveloper. The Oracle ADF framework can be used to implement enterprise solutions that search, display, create, modify, and validate data using web, wireless, desktop, or web services interfaces [12].

Oracle ADF is based on the MVC design pattern. Utilizing the Model-View-Controller design pattern, application development and maintenance requirements are minimized. Oracle ADF framework provides solutions for the individual Model-View-Controller layers and integrates the layers together with customization and security [13]. Oracle ADF makes it simple to develop agile applications creating a user experience that is as straightforward as dragging-and-dropping the desired business service onto a visual page designer within JDeveloper. The integrated solutions cover areas such as: Object/Relational mapping, data persistence, reusable Controller Layer, rich Web user interface framework, and data binding to user interface. Oracle ADF is also integrated with the Oracle Service Oriented Architecture (SOA) and WebCenter Portal framework, which simplifies the creation of complete composite applications [8]. MVC makes model classes reusable without additional modifications. MVC patterning can be very complex and difficult to implement in smaller software applications. The alternative to MVC is Object-Oriented design, which is the process of planning a system of interacting objects for the purpose of solving a software problem.

2. Oracle ADF Architecture

Oracle ADF is based on the Model-View-Controller pattern explained in the previous section and separates the Model Layer from the business services. This enables Service Oriented Architecture (SOA) development of applications. Therefore, the Oracle ADF architecture has four layers. The four layers are: Business Service Layer, Model Layer, Controller Layer, and View Layer. The Model Layer holds and integrates the many elements of Java EE. Figure 14 illustrates the four layers and developers options for which technology they could use in each layer when developing ADF applications.

Figure 14. Oracle ADF Architecture Layers, from [12].

a. Business Services Layer

The Business Service Layer is the component that mediates between an MVC application and a data source. The Business Services layer is responsible for retrieving data, represent the data as Java objects, and Implement business rules. Execution of the Business Services layer is streamlined by ADF Business Components. ADF Business Components create reusable data-aware business services, which minimizes developer coding. Visual editors and wizards are used to create ADF business Components without having to write the applications Java code. The ADF Business Components consist of five main components: entity objects, entity associations, view objects, view links, and application modules [14]. Figure 15 shows the relationship between the ADF Business Components.

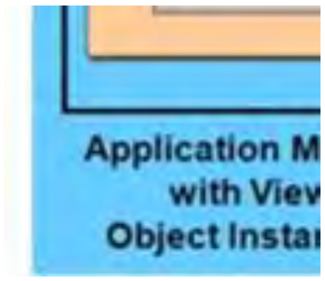


Figure 15. ADF Business Components: Entity objects, Entity associations, View objects, View Links,and Application modules, from [14].

- Entity Objects

ADF Entity objects can represent any business component that captures data as shown in Figure 16. Entity objects map to single objects in the data source such as tables, views, or snapshots in a database. Examples of Entity objects are: regions, divisions, departments, sections, sales, invoices, employees, or equipment.

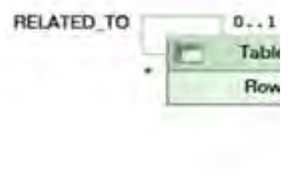


Figure 16. Entity Object within the ADF Business Component Architecture, from [14].

- View Objects

ADF view objects are business components that collect data from the data source, shape that data for use by clients, and allow clients to change that data in the ADF Business Components cache [9]. The View Layer represents the user interface of the application and its main role is to give an application-specific view of records queried into underlying entity objects. View Objects map to Entity Objects which embed SQL to filter or manipulate the data in the Entity Objects. As shown in Figure 17, the view object shapes and defines the data for an application by defining an SQL statement that selects only the attributes associated with the underlying entity objects. View objects can be read-only, one, or many-entity objects [9]. In order to manipulate data, one must create an ADF Business Components entity object that accesses and updates the data.

Figure 17. View Object within the ADF BC Architecture, from [14].

- Application Modules

The Application Module is the final step in arranging those view objects in a data model. This data model view objects is an application module and represents particular application tasks. The typical application model will contain one or more application modules. The application provides data model for task by aggregating the view objects for the task as shown in Figure 18. Application Modules can be used in three ways: service object, reusable object for nesting, or shared application module [14]. Examples of an application module are: updating customer information or creating new orders.

Figure 18. Application Module within the ADF Business Component Architecture, from [14].

b. Model Layer

The ADF Model is a declarative framework and central part of ADF. A Declarative framework program is declarative if it describes what something is like, rather than how to create it. This layer enables the user to create ADF applications based on different types of business services. The Model layer is between the business services and view and controller layers. The Model layer standardizes the interactions between each layer. ADF Model consists of two key components [14]:

- **Data controls**: Abstract the implementation technology of a business service by using standard metadata interfaces to describe the service's operations and data collections, including information about the properties, methods, and types involved.

- **Declarative bindings:** Used to bind services that are exposed by data controls to user interface components. They provide a way to call from the View Layer into the Model Layer using code.

c. Controller Layer

The Controller layer manages the applications' flow and handles user inputs. The Controller's key feature is the task flow. Task flows provide a modular approach for defining control flow in a Fusion web application. Instead of representing an application as a single large JSF page flow, it is broken up into multiple task flows, each of which contains a portion of the application's navigational graph [14].

Task flows consist of activity nodes, which are operations such as displaying a page or executing application logic. Task flows can be created visually using the diagram editor in JDeveloper. There are two types of task flows: bounded and unbounded. A fusion web application consists of a single ADF unbounded task flow and several bounded task flows. An unbounded task flow is the parent task flow from which an application is launched and any page within the task flow can be the starting page of the application [14]. As shown in Figure 19, a bounded task flow represents a reusable application flow that can be referenced from any other task flow. Bounded task flows have a single point entry and zero or more exit points, and they contain their own set of private control flows rules, activities, and managed beans [14].

Figure 19. ADF Bounded Workflow, from [14].

d. View Layer

The ADF View Layer can be used to take an application to a higher level with an assortment of different user interfaces. The View Layer represents the user interface of the application. In this layer, Oracle uses ADF Faces framework to build rich user interfaces. The ADF faces is a set of more than 150 Rich Internet Application (RIA) components built on top of the standard Java Server Faces (JSF) application programming interfaces (API), that use the newest technologies to provide interactive user interface. Benefits of using ADF Face are: large set of fully featured rich components, built in Ajax support, and limited need for developer to write JavaScript [14]. The categories of key ADF Faces components are:

- General Controls
  - Navigation components: Page navigation, buttons, and links
  - Images and Icons: From pictures to video
- Text and Selection components
  - Output components: Display text, graphics, and icons for audio/video clips
  - Input components: Allows users to enter data and upload files
- Data Views
  - Table and Tree components: Sort and filter capability
  - Data Visualization Components: Includes graphs, pivot tables, Gantt Charts, maps, and timelines
  - Query Components: Support multiple search criteria, adding and deleting criteria, selectable search operators, and match all/any selections
- Layout components
  - Containers to determine the layout of the page; includes interactive component that can show or hide content and provide sections or lists

3. Oracle ADF Benefits

Oracle ADF offers key benefits while developing web applications. The key features that make it attractive are:

- **End-to-End Solution:** Oracle ADF does not focus on just one layer of the Java EE architecture. ADF provides an integrated and complete solution for every Java EE layer from the View Layer and data-bindings through the business services and data access; it also supports every development life-cycle phase from inception through support [12].

- **Development Environment:** Oracle ADF has integrated support. Oracle JDeveloper provides visual aid and a declarative approach to minimize writing framework code [12].

- **Platform Independence:** Oracle ADF runtime is open source and can be installed on various Java EE-compliant application servers. Business services can connect to any SQL-92-compliant database [12].

- **Technology Choice:** Oracle ADF supports multiple technologies for each of the layers of the application and does not enforce a specific technology or a specific development style on the developer [12].

- **Technology Commitment:** Oracle ADF is the technology choice for the Oracle next generation set of enterprise applications. The product is used to develop portal applications, wireless applications, and web applications; therefore it provides a committed, supported, and consistent technology stack [12].

- **Metadata-Driven:** Oracle ADF framework offers declarative options for development configured from XML metadata, while accommodating custom coding wherever necessary. Users can choose to use all or part of the framework in the applications they build, making the application components much more reusable and flexible. The use of metadata also enables rules for data bound fields to be specified at the Model Layer. Labels, validation, and tooltip properties can be specified in the metadata for ADF data bindings—those properties are utilized independent of the user interface implementation [12].

- **Declarative Customization:** Oracle ADF provides a unique solution that allows an organization to use a single base application and customize it to fit the requirements of different users [12].

- **Enhanced Reusability:** Oracle JDeveloper and Oracle ADF provide support for superior reusability features including: JSF templating, reusable task flows, task flow templating, reusable business services, ADF libraries, and JSF fragment based regions [12].

- **Source availability:** Oracle provides the source code for the ADF framework to customers with a support license. Having the source available can help developers understand the underlying mechanisms of the framework and debug problems in their applications [12].

- **Support and Training:** Oracle provides around the clock support from an established organization and provides training through Oracle University which offers regular instructor-lead courses on Oracle ADF and JDeveloper [12].

E. ORACLE JDEVELOPER

Oracle JDeveloper is a free tool that enables developers to build standard-based enterprise applications. This free Oracle integrated development environment (IDE) is the main platform for developing fusion applications within Oracle. JDeveloper integrates development features for Java, HTML5, SOA, Web, Mobile, Database, XML, and Web services which share structure into a single development environment. JDeveloper is open source and can be operated on Windows, Linux, Mac OS X, and other UNIX-based systems [15]. JDeveloper is a development environment that reduces the learning curve of developers by simplifying the development process using a visual and declarative development approach. JDeveloper provides the developer a visual concept for tasks such as page flow design or page layout without preventing access to the underlying source code. If the developer wishes, JDeveloper allows access to the source code. JDeveloper eliminates tedious coding by using property inspectors, structure panes, and visual editors to define components of an application. Developers can design, generate, and visualize their codes with Unified Modeling Language (UML), Java, and database diagrams [16].

Oracle JDeveloper covers the full development lifecycle from initial analysis, design, coding, testing, and all the way to deployment. A WebLogic server is integrated into JDeveloper to test, operate, and debug the application within the development environment for Oracle databases as well as non-Oracle databases [15].

Figure 20 is a screenshot of Oracle JDeveloper and the most common editor windows and tools. The general purpose of each window is explained next [17].

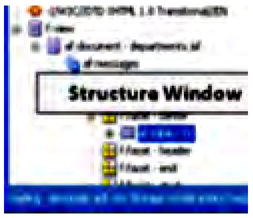


Figure 20. Screenshot of Oracle JDeveloper workspace, from [17].

- **Application Navigator:** This window is used to create, edit, find files for one's application (Project panel), and to interact with the database and other connections. The application navigator helps manage the contents and associated resources of an application [18].

- **Application Resource Panel**: This window displays the application-level resources and configuration files. It offers access to connections for the currently selected application (database or application server) and descriptor files that are used to configure the application. This includes database connection information metadata files used to configure ADF Business Components [17].

- **Data Control Panel:** The data control panel displays the data collections, attributes, built-in operations, and business methods from the business services exposed through a data control registry. Items can be dragged and dropped from the data control panel on the UI. This generates a metadata XML file to bind the business data with the UI [17].

- **Structure Window:** The structure window displays a structural view of the data in the document that is currently selected and displays icons identifying type of object in the active window. Structure window can be used to view or reorder the source code using drag and drop components from any palette to the structure window [17].

- **Visual Editor:** The visual editor window will help to visually build the UI for ADF applications. The visual editor allows developers to visually lay out the UI. JDeveloper synchronizes the selection in the structure window with the visual editor and vice versa [17].

- **Component Palette:** The component palette window lists down available components for designing a page. Components are associated with the selected technology (Java Client, JSFJSP files, or HTML files) that is being utilized to design pages or for defining navigation [17].

- **Property Inspector**: A property is a named attribute of a class or component that can affect its appearance or its behavior. The property inspector displays the exposed properties of the component selected in the structure window or in the visual editor [17].

- **Log Window**: The log window displays the logs from various components such as compiler, audit rules, debugger, and profiler [17].

F. ORACLE WEBLOGIC SERVER

The purpose of the Oracle WebLogic Server is to deploy the web application so end users can access the applications using a web browser. The WebLogic server is a scalable, web application server which implements the Java enterprise edition standards. The WebLogic server supports many types of distributed applications and is an ideal foundation for building applications based on service-oriented architectures. The WebLogic server complete implementation of the Java EE provides a standard set of application programming interfaces that create Java applications that can access databases, email, and connections to external enterprise systems [19]. The basic concepts of a WebLogic Server are:

- **WebLogic Domain Structure:** A domain is a logically related group of WebLogic Server resources that is managed as a unit. There is a minimum of one administration server per domain; all other servers are called managed servers. The administration server is the central point of managing the WebLogic server domain [20].
- **Managed Servers:** These are additional WebLogic servers that deploy Web application, Enterprise Java Beans, Web Services, and associated resources. When a managed server starts, it synchronizes to the admin server to optimize performance [20].
- **WebLogic Server Cluster:** A cluster is a group of WebLogic Server instances that work together to increase performance and to provide scalability and availability for applications [20].

G. ORACLE SQL DEVELOPER

Oracle SQL Developer is a free graphical user interface integrated development environment that simplifies database management and SQL development tasks. SQL Developer can be run on Windows, Linux, and Mac OS X. This productivity tool streamlines SQL development time and offers complete end-to-end development of your PL/SQL applications. Developers can debug and edit PL/SQL statements, create, modify and browse objects, run SQL statements, run queries, and export data to a desired format such as XML, Excel, HTML, and PDF [21].

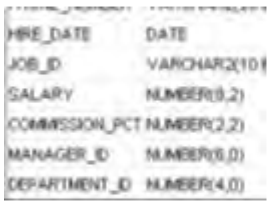
The Key features of SQL developer are:

- **Managing Connections:** A simple wizard is used in SQL Developer to create new database connections as shown in figure 21. Once connected, the developer can browse the database, run reports, create schema objects and create, execute, and debug PL/SQL [21].

Figure 21. SQL Developer New database connection, from [21].

- **Working with the SQL Worksheet:** The SQL worksheet opens when connection is established. Developers can create and run SQL commands individually or together in the worksheet. Developers can specify actions proceeded by the database such as selecting data from a table, inserting data, creating a table, and saving data to file [21].

- **Browsing the Database:** The connection navigator allows the developer to browse through objects in the database schema including tables, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views and materialized view logs, synonyms and public synonyms, database links, and directories as shown in figure 22 This allows the developer to quickly retrieve information as it is displayed in an easy to read tab window [21].



[illegible]	[illegible]
HIRE_DATE	DATE
JOB_ID	VARCHAR2(10
SALARY	NUMBER(8,2)
COMMISSION_PCT	NUMBER(2,2)
MANAGER_ID	NUMBER(6,0)
DEPARTMENT_ID	NUMBER(4,0)

Figure 22. SQL Developer browsing the database with the table viewer, from [21].

- **Producing SQL Scripts:** The connections navigator allows the editing and updating of database objects. As the developer creates new objects or edits existing ones, the data definition language (DDL) is available for review [21].

- **PL/SQL Development:** This includes a full-featured editor for PL/SQL program units, custom PL/SQL syntax highlighting, and editing functions such as bookmarks, code completion, code folding, and search and replace [21].

- **Database Reporting:** SQL Developer provides many predefined reports about the database and objects. SQL Developer allows developers to create and save reports for repeated use [21].

H. SUMMARY

In this chapter, we discussed the development approach and described Oracle products utilized in the development of XXMC reports proof-of-concept. In Chapter V, we will describe the development process and describe the capability, functionality, and features of the XXMC Reports.

V. APPLICATION DEVELOPMENT

A. INTRODUCTION

In the previous chapters, we discussed the purpose of GCSS-MC Use Cases that were analyzed, and we described the tools used for its design and development. This chapter describes the ADF application implementation: how the application user interface looks and what it can and cannot do. We first describe the implementation process and then we discuss the application features and functionalities, as well as some considerations related to the user interfaces and application functions.

B. DEVELOPMENT PROCESS

As described in Chapter IV, numerous Oracle tools were used to develop the XXMC Dashboard. The tasks performed using these tools are described in the following two sections. The first section addresses the database implementation, and the second discusses the process implementation.

C. APPLICATION DEVELOPMENT

In the design and development of the XXMC Reports Dashboard application, there were several factors that were taken into consideration. The user interface needed to be user friendly, easily understood, and graphically simple. The design was created with an attempt to make user interaction easy to navigate between all the Use Cases that were analyzed and developed. We included search functions and drop down boxes to eliminate the need for users to memorize all the different criteria that were searchable.

Figure 23 is a snapshot of the XXMC Reports Dashboard Homepage. This page has six buttons that corresponds to each individual Use Case. Each page can be opened simultaneously on separate tabs to run concurrently, or each can be run individually.

Figure 23. Readiness Report (Alpha) TAMCN Class.

The XXMC Reports Dashboard Homepage is utilized to navigate between pages; it provides no ability to modify any information that is being accessed through the XXMC R001 database that is being utilized to pull information for analytics. The following section will provide a scenario of each Use Case to demonstrate the functionality of this proof-of-concept.

D. FUNCTIONALITIES/SCENARIOS

Understanding the functionality of the application is best understood by utilizing Use Case scenarios. In this section, we will describe all six Use Cases, the steps required, and the products produced. These Use Cases can be used by Marines at all levels of Supply and Maintenance.

1. Readiness Report (Alpha) Use Case

This Use Case provides the user with the ability to pull maintenance readiness numbers for individual units (Figure 24). When a user accesses the Alpha Use Case page, he has the ability to search by Activity Address Code which is unique to each unit in the Marine Corps. Once this page displays all requested information, the user is unable to edit any information that is presented. This page also contains six navigation buttons that can be utilized to return to the Dashboard homepage or to navigate directly to another Use Case.

A user will access the XXMC Reports Dashboard Homepage and select Use Case Alpha. After clicking on the Alpha Tab, the user simply inputs his or her Activity Address Code and then clicks the search button to execute the search. The user has the ability to reset the search page by clicking on the reset button. Figure 24 displays the search pane for Readiness Report (Alpha).

Figure 24. Readiness Report (Alpha) Activity Address Code Search.

Once the user queries the readiness report by clicking on the search button, the analytics will be displayed (Figure 25).The initial view displays the aggregated unit readiness by TAMCN Class (Figure 25). This view shows the unit's aggregated readiness percentage which is calculated by dividing total deadline by total on hand. The user now has the ability to select any TAMCN Class. Then, the information will drill down to show each individual TAMCN associated with the above TAMCN Class. This information is displayed to show each major end item and its readiness rating (Figure 26).

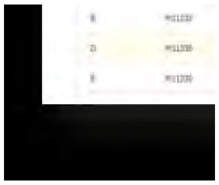


Figure 25. Readiness Report (Alpha) TAMCN Class View.

Figure 26. Readiness Report (Alpha) Individual TAMCN View.

Once this information is displayed, each individual TAMCN number under a selected TAMCN class and its Readiness percentage is illustrated. Users are able to further drill down by clicking on a TAMCN that has a deadline number greater than zero. Once a user clicks on a selected TAMCN, the information displayed will contain all NSNs associated with the selected TAMCN (Figure 27).

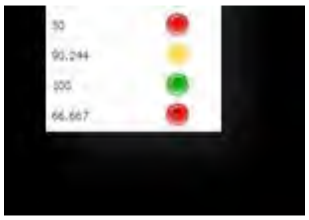
This view displays all required information concerning each deadlined item by NSN and Serial Number.

Figure 27. Readiness Report (Alpha) TAMCN NSN View.

Users are able to display all deadlined items for a selected Activity Address Code by clicking on the Deadline Items tab once the initial search of Activity Address Code is performed. This view will display all deadlined items sorted by TAMCN class and NSN for a rollup of all equipment within a selected unit (Figure 28).

Figure 28. Readiness Report (Alpha) Deadlines Items View.

Use Case Readiness Report (Alpha) utilizes gauges on all views of deadlined equipment. These gauges were placed on overall readiness levels: Green Greater than 95%, Yellow 85% to 94%, and Red Under 85% (Figure 29). This readiness percentage can be modified to fall in line with appropriate Marine Corps Orders and directives.



50	🔴
90.244	🟡
100	🟢
66.667	🔴

Figure 29. Readiness Report (Alpha) Gauge View.

2. Maintenance Effort Report (Bravo) Use Case

This Use Case is designed to allow users to search for any Activity Address Code within GCSS-MC and, additionally, specific Service Request Types. This search will be utilized to manage repair times for equipment inducted into the maintenance cycle. Over time it can be analyzed to determine trends in man-hours utilized to repair end items.

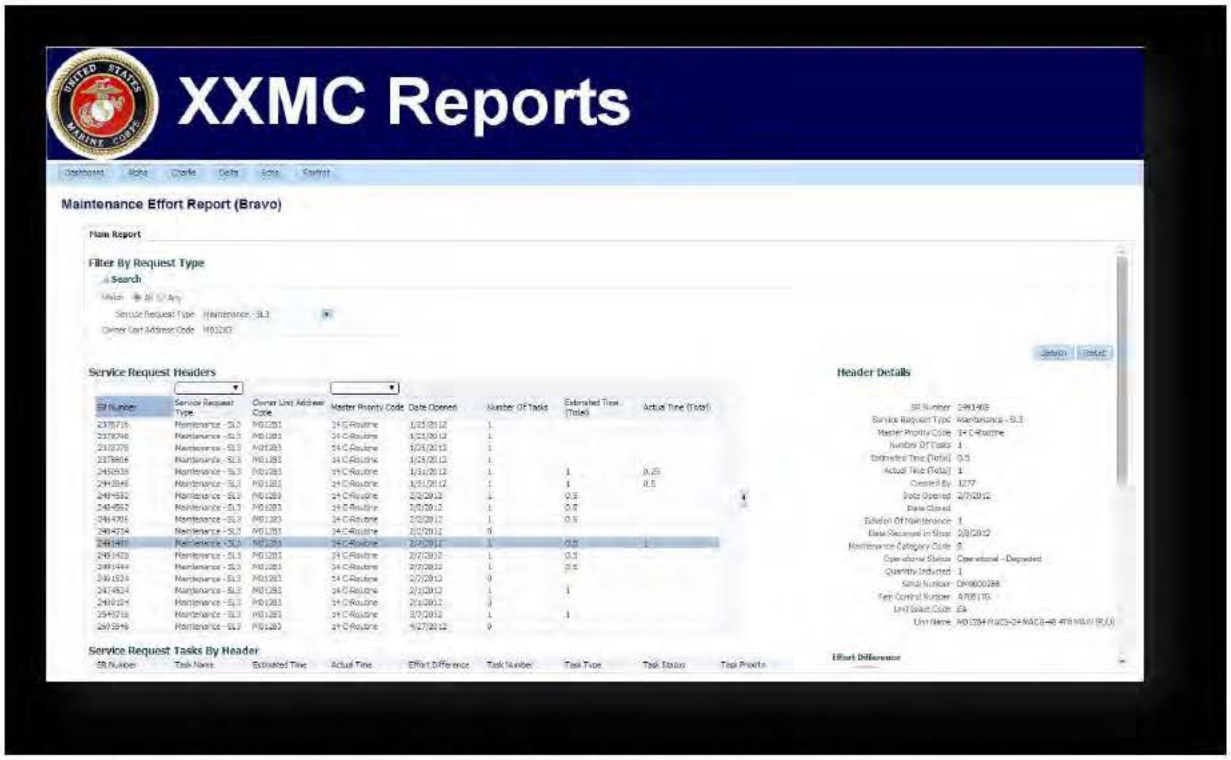
Once the user accessed the Maintenance Effort Report (Bravo) search page, the user is able to enter any Activity Address Code and select from a pre-determined drop down box of Maintenance types (Figure 30).

Filter By Request Type

Search

Figure 30. Maintenance Effort Report (Bravo) Search View.

Upon submission of Activity Address Code and Service request type, the query will be run and the information will be displayed in the format in Figures (31), (32), and (33).



Maintenance Effort Report (Bravo)

Main Report

Filter By Request Type

Search

Match [✓ All] Any

Service Request Type Maintenance - SL3

Owner Unit Address Code M01283

Service Request Headers

SR Number	Service Request Type	Owner Unit Address Code	Master Priority Code	Date Opened	Number Of Tasks	Estimated Time (Total)	Actual Time (Total)
2378716	Maintenance - SL3	M01283	14 C-Routine	1/25/2012	1		
2378748	Maintenance - SL3	M01283	14 C-Routine	1/25/2012	1		
2378778	Maintenance - SL3	M01283	14 C-Routine	1/25/2012	1		
2378806	Maintenance - SL3	M01283	14 C-Routine	1/25/2012	1		
2450938	Maintenance - SL3	M01283	14 C-Routine	1/31/2012	1	1	0.25
2443848	Maintenance - SL3	M01283	14 C-Routine	1/31/2012	1	1	0.5
2484532	Maintenance - SL3	M01283	14 C-Routine	2/2/2012	1	0.5	
2484562	Maintenance - SL3	M01283	14 C-Routine	2/2/2012	1	0.5	
2484706	Maintenance - SL3	M01283	14 C-Routine	2/2/2012	1	0.5	
2484734	Maintenance - SL3	M01283	14 C-Routine	2/2/2012	0		
2491408	Maintenance - SL3	M01283	14 C-Routine	2/7/2012	1	0.5	1
2491428	Maintenance - SL3	M01283	14 C-Routine	2/7/2012	1	0.5	
2491444	Maintenance - SL3	M01283	14 C-Routine	2/7/2012	1	0.5	
2491524	Maintenance - SL3	M01283	14 C-Routine	2/7/2012	0		
2474524	Maintenance - SL3	M01283	14 C-Routine	2/1/2012	1	1	
2480124	Maintenance - SL3	M01283	14 C-Routine	2/1/2012	0		
2549716	Maintenance - SL3	M01283	14 C-Routine	3/7/2012	1	1	
2695948	Maintenance - SL3	M01283	14 C-Routine	4/27/2012	0		

Header Details

SR Number	2491408
Service Request Type	Maintenance - SL3
Master Priority Code	14 C-Routine
Number Of Tasks	1
Estimated Time (Total)	0.5
Actual Time (Total)	1
Created By	1277
Date Opened	2/7/2012
Date Closed	
Echelon Of Maintenance	1
Date Received In Shop	2/8/2012
Maintenance Category Code	8
Operational Status	Operational - Degraded
Quantity Inducted	1
Serial Number	DM9000288
Item Control Number	A7081TG
Unit Issue Code	EA
Unit Name	M01284 MACS-24 MACG-48 4TH MAW (RUU)

Service Request Tasks By Header

SR Number	Task Name	Estimated Time	Actual Time	Effort Difference	Task Number	Task Type	Task Status	Task Priority

Effort Difference

Figure 31. Maintenance Effort Report (Bravo) Information View Top.

[redacted]

Figure 32. Maintenance Effort Report (Bravo) Information View Bottom.

Figure 33. Maintenance Effort Report (Bravo) Information View Side.

Maintenance Effort Report (Bravo) allows the user to individually select a Service Request (SR) Number to see all associated maintenance history and the estimated and actual effort-hours associated to each service request. This information will allow quick analysis of historical trends of equipment that is exceeding estimated hours and can allow for new procedures to be put into place to fix any discrepancies with man-hours. All gauges that are utilized can be customized during the initial creation of the page template.

3. Secondary Reparable Report (Charlie) Use Case

This Use Case shows trends in faulty secondary reparable parts that are fixed at different echelons of maintenance. These parts consist of items such as generators, engines, and circuit boards that typically cannot be completed at the primary user level of maintenance.

Users are required to enter the Activity Address code to search for items assigned to their unit. If they have criteria readily available, users can also search by a TAMCN number or Record NSN to limit their search (Figure 34).

Figure 34. Secondary Reparables Report (Charlie) Search View.

Upon submission of the search criteria, the information will be displayed in the format in Figures 35 and 36.

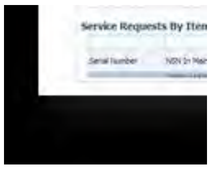


Figure 35. Secondary Reparables Report (Charlie) Information View Top.

Figure 36. Secondary Reparables Report (Charlie) Information View Bottom.

This information is displayed so users can individually select a Record NSN to see all information available. The bottom portion of the view will show each service request number for a Record NSN that has had repairs completed. This Use Case allows for trend analysis of faulty Secondary Reparables as well as man-hours required to complete these repairs. As with the Maintenance Effort Report (Bravo), these analytics can be utilized to change maintenance procedures when dealing with the repair times of Secondary Reparables.

4. Cost Report (Delta) Use Case

This page allows the user to pull financial information in several ways. There are two tabs that users can select from the Delta page. The main report tab and unit totals tab. From the main report tab, a user can pull information by searching for a service request number or a Record NSN. Users may also search by individual unit or by regional activity such as an MSC and MEF. From the unit totals tab, users can search for an individual unit or all units—the costs are displayed. This report will be utilized to observe how resources are being expended at every level. Over time, analytics can be developed to reveal what products are driving command’s overall costs.

Upon submission of any of the above search criteria, the information will be displayed in the format in Figures 37 and 38.

8371984	Supply
7835016	Supply
7871172	Supply
5113928	Supply

Repair Parts By Service Request

Figure 37. Cost Report (Delta) search by Regional Activity Code View Top.

7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC
7855468	MC

Figure 38. Cost Report (Delta) search by Regional Activity Code View Bottom.

The information is displayed so users can easily select any record in the top view. When a user selects a service number row in the top view, he will see that the bottom view is automatically populated with all financial information for that service request. A pie chart is automatically created; it depicts a breakdown of cost by NSN as shown in Figure 39.

Figure 39. Cost Report (Delta) Breakdown of Cost by NSN.

Figure 40 displays what the user will see when he selects both the unit totals tab and a regional activity. The top pie chart displays all the regional activities and their percentages of the total repair parts' cost. The bottom pie chart displays all of the subordinate commands within the regional activities and their percentages of the total costs of that regional command.

MMFAH8	MMT300
M05174	MMT300
M11204	MMT300
M13510	MMT300
M35100	MMT300
M38033	MMT300

Figure 40. Cost Report (Delta) Unit Totals Tab.

5. GCSS-MC Maintenance Production Report (Echo) Use Case

The Maintenance Production Report is designed to view maintenance-related information. This page has three tabs: Main Report, Tasks & Orders, and Order Status Details. A user may search all service requests by unit activity code or further filter the service requests by type of service request (examples: Supply, Maintenance, or Calibration). Figure 41 displays the main report and what the user will see when filtering by unit activity code and by supply service request.

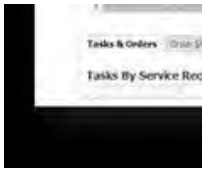


Figure 41. Maintenance Production Report (Echo) View Top.

When a user selects any row, the information about that service request will be displayed within the Tasks & Orders tab on the bottom half of the page. The information is displayed so the user can see the tasks by service request. The user then may select a task by service request; then all order information will be displayed pertaining to that task on the bottom right side of the page as shown in Figure 42.

Figure 42. Maintenance Production Report (Echo) Tasks & Orders Tab Bottom Page View.

The Order Status Detail tab allows users to view the orders by service request. Once a user selects a row on the main report, the orders for that particular service request will automatically populate in the bottom view of the page along with the entire order status history for that particular document number. Figure 43 displays how this page will appear.

Figure 43. Maintenance Production Report (Echo) Orders Status Details Tab Bottom Page View.

6. Document Number Report (Foxtrot)

This page is designed to allow the user to view all outstanding and fulfilled repair part requisitions for open and completed service requests. This page has two tabs: Main Report and the Orders by NSN. These tabs allow the users to select how they want to conduct their searches for document numbers. The main report allows the user to search service request data based on the unit activity of the service request, the Record NSN, or the document number associated with an order. As shown in Figure 44, the table displays all service requests that match the user's search.

2545968	Suppl
2546220	Suppl
2546622	Suppl
2546708	Suppl
2548542	Suppl
2548814	Suppl
2550272	Suppl

Figure 44. Document Report Number (Foxtrot) Main Report View.

A user may select a service request and, then, the details of that service request will populate on the bottom half of the page. All document numbers associated with a service request will be shown whether or not the requisition has been received. A status indicator also shows the percentage of completed orders for a service request. Figure 45 shows how the page will be displayed.

Figure 45. Document Report Number (Foxtrot) Main Report Bottom Page View.

The user may select the Orders by NSN tab to display repair part requisitions directly without having to navigate through a specific service request. The user can search based on the organization that will be receiving the part or based on the Record NSN of the part. The Orders by NSN page is depicted in Figure 46.

Figure 46. Document Report Number (Foxtrot) Orders by NSN Page View.

VI. CONCLUSION

This chapter summarizes this thesis' research including the proof-of-concept prototype, analysis, design, and implementation of the XXMC dashboard. There are several lessons learned that will benefit follow-on research and future development for GCSS-MC business analytics. This chapter also recommends potential research opportunities for the Marine Corps.

This chapter is organized as follows: Section A summarizes this thesis; Section B offers lessons learned; and Section C proposes future research opportunities.

A. SUMMARY

This thesis discussed the need for commanders at all levels of command to have rapid access to status reports and logistical data in order to make decisions that will enhance their unit readiness. We also discussed the way that current GCSS-MC Reports are generated utilizing the current Discovery Report Viewer application. Although some information can be generated and analyzed utilizing Discovery Report viewer, this information cannot be analyzed without being exported to external software. This process becomes time consuming and requires additional manipulation of the data retrieved. We believe that the XXMC Reports, the prototype application we designed and built, has the ability to pull and analyze any data that it can access. The ADF Framework that we utilized allows for manipulation of requested information by allowing trained individuals to modify the template that was generated. Additional Use Cases can also be created to analyze new requirements or refinements to the Use Cases used in this thesis.

The requirements' analysis was divided into two main parts: the XXMC R001 Database and the Use Case requirements. The XXMC R001 Database was required to be modified to normalize data utilized during the Use Case. The SQL scripts required to complete this are found in Appendix A. Next, the Use

Cases were analyzed to identify what fields from what XXMC R001 table were required to generate the analytics produced through the XXMC Dashboard. Once all Use Cases were analyzed, we were able to begin creating the relationships utilizing the ADF framework to create the prototype.

After the requirements' validation, we developed XXMC Dashboard using Model View Controllers as the design pattern and Oracle ADF, which implements MVCs as the development framework. We utilized many Oracle tools in the application of this prototype. Oracle SQL Developer Data Modeler, and Oracle SQL Developer were used for the database modeling and implementation. Oracle JDeveloper is the main Integrated Development Environment (IDE), and it was used to develop and deploy (using the integrated WebLogic server) the application for testing. The use of these tools allowed for the implantation of the final XXMC Dashboard prototype.

The user interface of the application was designed to reflect a single dashboard and six individual pages that identify each Use Case. The layout was created for ease of use and allowed for quick transaction between Use Cases and dashboard. The search function that was implemented on each individual Use Case page allows users to quickly filter through information contained in the XXMC R001 database. Drop down boxes were created with a pre-defined set of scenarios that are typically analyzed for maintenance reports.

B. LESSONS LEARNED

This section summarizes the lessons learned during this thesis development that will benefit future researchers in this area. Lessons learned are:

- Working with the sponsor to provide detailed Use Cases ensured a focused development effort and provided us with an understanding of what analytics the sponsor imagined.
- The database received from GCSS-MC had several duplicate and orphan entries that had to be cleaned out before developing the web application in JDeveloper.

- There was incomplete data in several key database fields (example: actual man-hours on work requests were not filled out and left blank). These fields can be required entries in order to process a service request using Oracle Forms.

- There were no primary or foreign keys in the R001 database extract received from GCSS-MC. These foreign keys are present in the complete GCSS-MC database and, since the R001 is an extract, the foreign keys are removed.

- Getting Oracle training via online courses was very helpful in learning the development process. Many other materials are available through the Oracle website, Oracle books, and forums.

- Since a large portion of building an application is modifying and exposing database tables, having minimum knowledge of basic SQL and databases would help with the developing experience.

- JDeveloper features can be extended or overwritten using Java. Thus, having someone proficient in Java would help development.

- Computers with at least 8 GB of RAM and a high-speed processor would increase the development efficiency.

C. FUTURE AREAS OF RESEARCH

In our research, we conducted a system requirements' analysis and designed a proof-of-concept XXMC prototype dashboard that is the initial stage in building readily available analytical reports from Marine Corps supply and logistical data. This relevant work can be broadened through additional research to many areas. Recommend areas of future research are security, fleet-wide XXMC implementation, and additional analytical requirements. We now discuss these proposed research areas.

1. Security

An in-depth analysis of XXMC security requirements in deploying an XXMC application would be a beneficial research area. This research could consist of examining Oracle security features to determine the soundest security settings and policies for follow-on versions of XXMC dashboard.

2. XXMC Implementation

XXMC fleet-wide implementation was outside the range and scope of this thesis. The next step would be to actually deploy this application on a server in a controlled environment that can be remotely accessed by several authorized users. The users can provide feedback, asses the applications effectiveness, and guide adjustments for enhancements.

3. Additional Analytical Requirements' Study

A potential research area would provide detailed requirements' analysis that closely involves the GCSS-MC users and stakeholders. This analysis would document critical requirements, feedback, information, and concerns. This analysis could be used to design a future XXMC model that produces the precise logistical information that commanders desire.

APPENDIX A. SQL SCRIPT AND SCREENSHOTS

Figures 47 to 61 illustrate the View Object relationships and the tab design for all Use Cases analyzed.

A. READINESS REPORT (ALPHA)

1. View Objects

a. ATamClassVO

SELECT SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) AS TAMCN_CLASS, SUM(InventoryTblEO.QUANTITY_ONHAND) AS TOTAL_ONHAND, InventoryTblEO.ACTIVITY_ADDRESS_CODE FROM XXMC_R001_INVENTORY_TBL InventoryTblEO, XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHERE ItemmasterTblEO.TAMCN is not null and InventoryTblEO.RECORD_NSN = ItemmasterTblEO.RECORD_NSN and (ItemmasterTblEO.MARES_CATEGORY
= 'MARES MEE' or ItemmasterTblEO.MARES_CATEGORY = 'MARES NON-MEE') and InventoryTblEO.QUANTITY_ONHAND > 0
and SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) in ('A', 'B', 'C', 'D', 'E') and InventoryTblEO.ACTIVITY_ADDRESS_CODE is not null
GROUP BY SUBSTR(ItemmasterTblEO.TAMCN, 1, 1), InventoryTblEO.ACTIVITY_ADDRESS_CODE

b. ATamcnDisplayVO

SELECT SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) AS TAMCN_CLASS, SUM(InventoryTblEO.QUANTITY_ONHAND) AS TOTAL_ONHAND, InventoryTblEO.ACTIVITY_ADDRESS_CODE, itemmasterTblEO.TAMCN FROM XXMC_R001_INVENTORY_TBL InventoryTblEO, XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO WHERE ItemmasterTblEO.TAMCN is not null and InventoryTblEO.RECORD_NSN =ItemmasterTblEO.RECORD_NSNand(ItemmasterTblEO.MARES_CATE

```
GORY = 'MARES MEE' or ItemmasterTblEO.MARES_CATEGORY =
'MARES NON-MEE') and InventoryTblEO.QUANTITY_ONHAND > 0
and SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) in ('A', 'B', 'C', 'D', 'E') and
InventoryTblEO.ACTIVITY_ADDRESS_CODE is not null
GROUPBYItemmasterTblEO.TAMCN,InventoryTblEO.ACTIVITY_ADDRE
SS_CODE
```

c. AHeadersDisplayVO

```
SELECT SrheadersTblEO.BATCH_ID,
SrheadersTblEO.CREATED_BY,
SrheadersTblEO.CREATION_DATE,
SrheadersTblEO.DATE_CLOSED,
SrheadersTblEO.DATE_RECEIVED_IN_SHOP,
SrheadersTblEO.DEADLINED_DATE,
SrheadersTblEO.DEFECT_CODE,
SrheadersTblEO.ECHELON_OF_MAINT,
SrheadersTblEO.EQUIP_OPER_TIME_CODE,
SrheadersTblEO.ERROR_MESSAGE,
SrheadersTblEO.EXTERNAL_APPLICATION,
SrheadersTblEO.FLIGHT_STATUS,
SrheadersTblEO.HOLD_UNIT_IDENT_CODE,
SrheadersTblEO.ITEM_DESIGNATOR_NUMBER,
SrheadersTblEO.JOB_ORDER_NUMBER,
SrheadersTblEO.JOB_STATUS_CODE,
SrheadersTblEO.JOB_STATUS_DATE,
SrheadersTblEO.LAST_UPDATE_DATE,
SrheadersTblEO.LAST_UPDATED_BY,
SrheadersTblEO.MAINT_CATEGORY_CODE,
SrheadersTblEO.MASTER_PRIORITY_CODE,
SrheadersTblEO.METER_READING,
SrheadersTblEO.MILITARY_LABOR_HOURS,
SrheadersTblEO.NSN_IN_MAINTENANCE,
SrheadersTblEO.OPENED_DATE,
SrheadersTblEO.OPERATIONAL_STATUS,
SrheadersTblEO.OWNER_UNIT_ADDRESS_CODE,
SrheadersTblEO.PROBLEM_SUMMARY,
SrheadersTblEO.PROCESS_STATUS,
SrheadersTblEO.QUANTITY_INDUCTED,
SrheadersTblEO.RECORD_ID,
SrheadersTblEO.REGIONAL_ACTIVITY_CODE,
SrheadersTblEO.REQUEST_ID,
SrheadersTblEO.SERIAL_NUMBER,
SrheadersTblEO.SERVICE_REQUEST_TYPE,
SrheadersTblEO.SR_NUMBER,
SrheadersTblEO.SR_OPENED_BY,
SrheadersTblEO.SR_XREF,
ItemmasterTblEO.TAMCN,
SrheadersTblEO.TASK_NAME,
SrheadersTblEO.TOTAL_CIV_LAB_EXPENSE,
SrheadersTblEO.TOTAL_EQUIP_OPER_TIME,
SrheadersTblEO.UNIT_ISSUE_CODE,
SrheadersTblEO.UNIT_NAME,
SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) as TamClass
FROM  XXMC_R001_SRHEADERS_TBL SrheadersTblEO,
XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHEREItemmasterTblEO.RECORD_NSN=
SrheadersTblEO.NSN_IN_MAINTENANCE
and SrheadersTblEO.OPERATIONAL_STATUS = 'Deadlined' and
SrheadersTblEO.NSN_IN_MAINTENANCE in
(SELECT UNIQUE(InventoryTblEO.RECORD_NSN)
FROM XXMC_R001_INVENTORY_TBL InventoryTblEO,
```

XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHERE InventoryTblEO.RECORD_NSN =
ItemmasterTblEO.RECORD_NSN and ItemmasterTblEO.TAMCN
is not null and (ItemmasterTblEO.MARES_CATEGORY = 'MARES MEE'
or ItemmasterTblEO.MARES_CATEGORY = 'MARES NON-MEE')
and InventoryTblEO.QUANTITY_ONHAND > 0)

d. AHeadersUniqueVO

SELECT XxmcR001SrheadersTbl.NSN_IN_MAINTENANCE,
XxmcR001SrheadersTbl.SERIAL_NUMBER,
MAX(XxmcR001SrheadersTbl.QUANTITY_INDUCTED)asQuantity,
ItemmasterTblEO.TAMCN, SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) as
TamClass, XxmcR001SrheadersTbl.OWNER_UNIT_ADDRESS_CODE
FROM XXMC_R001_SRHEADERS_TBL XxmcR001SrheadersTbl,
XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHERE ItemmasterTblEO.RECORD_NSN=
XxmcR001SrheadersTbl.NSN_IN_MAINTENANCEandXxmcR001Srhead
ersTbl.OPERATIONAL_STATUS='Deadlined'and
XxmcR001SrheadersTbl.SERIAL_NUMBER is not null and
XxmcR001SrheadersTbl.NSN_IN_MAINTENANCE in
(SELECT UNIQUE(InventoryTblEO.RECORD_NSN)
FROM XXMC_R001_INVENTORY_TBL InventoryTblEO,
XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHERE InventoryTblEO.RECORD_NSN =
ItemmasterTblEO.RECORD_NSN and ItemmasterTblEO.TAMCN is not
null and (ItemmasterTblEO.MARES_CATEGORY = 'MARES MEE' or
ItemmasterTblEO.MARES_CATEGORY = 'MARES NON-MEE') and
InventoryTblEO.QUANTITY_ONHAND > 0)
GROUPBYXxmcR001SrheadersTbl.SERIAL_NUMBER,
XxmcR001SrheadersTbl.OWNER_UNIT_ADDRESS_CODE,

```
XxmcR001SrheadersTbl.NSN_IN_MAINTENANCE,
ItemmasterTblEO.TAMCN
union all    SELECT XxmcR001SrheadersTbl.NSN_IN_MAINTENANCE,
XxmcR001SrheadersTbl.SERIAL_NUMBER,
XxmcR001SrheadersTbl.QUANTITY_INDUCTEDasQuantity,
ItemmasterTblEO.TAMCN,SUBSTR(ItemmasterTblEO.TAMCN, 1, 1) as
TamClass,  XxmcR001SrheadersTbl.OWNER_UNIT_ADDRESS_CODE
FROM XXMC_R001_SRHEADERS_TBL XxmcR001SrheadersTbl,
XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHERE ItemmasterTblEO.RECORD_NSN=
XxmcR001SrheadersTbl.NSN_IN_MAINTENANCE and
XxmcR001SrheadersTbl.OPERATIONAL_STATUS = 'Deadlined' and
XxmcR001SrheadersTbl.SERIAL_NUMBER is null and
XxmcR001SrheadersTbl.NSN_IN_MAINTENANCE in
(SELECT UNIQUE(InventoryTblEO.RECORD_NSN)
FROM XXMC_R001_INVENTORY_TBL InventoryTblEO,
XXMC_R001_ITEMMASTER_TBL ItemmasterTblEO
WHERE InventoryTblEO.RECORD_NSN =
ItemmasterTblEO.RECORD_NSN and ItemmasterTblEO.TAMCN
is not null and (ItemmasterTblEO.MARES_CATEGORY =      'MARES
MEE' or ItemmasterTblEO.MARES_CATEGORY =      'MARES NON-
MEE') and InventoryTblEO.QUANTITY_ONHAND > 0)
```

2. Transient Attributes

a. Total Deadlined

Groovy expression:

```
if(adf.object.AHeadersUnique.sum("(Quantity == null || Quantity == 0) ? 1 :
Quantity") == null) {0} else {adf.object.AHeadersUnique.sum("(Quantity ==
null || Quantity == 0) ? 1 : Quantity")}
```

b. Total Deadlined (2)

Groovy expression:

if(adf.object.AHeadersUnique.sum("(Quantity == null || Quantity == 0) ? 1: Quantity") == null) {0} else {adf.object.AHeadersUnique.sum("(Quantity == null || Quantity == 0) ? 1: Quantity")}

c. Readiness Percentage

Groovy expression:

if(TotalDeadlined != null && TotalOnhand != null) {(1-(TotalDeadlined/TotalOnhand))*100} else {null}

d. Readiness Percentage (2)

Groovy expression:

if(TotalDeadlined != null && TotalOnhand != null){(1-(TotalDeadlined/TotalOnhand))*100} else {null}

3. View Object and Page Display

Figure 47. Readiness Report (Alpha) View Objects.

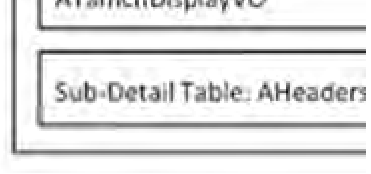


Figure 48. Readiness Report (Alpha) Tab Design 1.

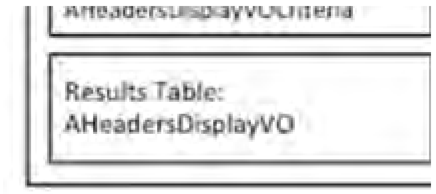


Figure 49. Readiness Report (Alpha) Tab Design 2.

B. MAINTENANCE EFFORT REPORT (BRAVO)

1. View Objects

a. BHeadersViewVO

SELECT *FROM NOTNULLATTEMPT BHeadersViewEO
WHERE BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – CAL' or BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – MOD' or BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – CM' or BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – PM' or BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – MISC' or BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – SL3' or BHeadersViewEO.SERVICE_REQUEST_TYPE = 'Maintenance – SRP'
SELECT *FROM XXMC_R001_SRHEADERS_TBL headers
Where headers.SR_NUMBER IN(
SELECT t1.SR_NUMBER FROM (SELECT h.SR_NUMBER , count(t.TASK_NUMBER) AS NUMTASKS
FROMXXMC_R001_SRHEADERS_TBLh, XXMC_R001_SRTASKS_TBLt
WHEREh.SR_NUMBER=t.INCIDENT_NUMBERANDt.ACTUAL_EFFORT is not null AND t.planned_effort is not null GROUP BY h.SR_NUMBER) t1,(SELECT h.SR_NUMBER , count(t.TASK_NUMBER) AS NUMTASKS
FROMXXMC_R001_SRHEADERS_TBLh, XXMC_R001_SRTASKS_TBLt

WHERE h.SR_NUMBER = t.INCIDENT_NUMBERGROUP BY h.SR_NUMBER) t2 WHERE t1.SR_NUMBER = t2.SR_NUMBER and t1.NUMTASKS = t2.NUMTASKS)

b. SrTasksTblVO

SELECT* FROM XXMC_R001_SRTASKS_TBLXxmcR001Srtasksbl

2. Transient Attributes

a. ActualEffortHours

Groovy expression:

if(ActualEffortUom == "MN"){ActualEffort*720} else if(ActualEffortUom == "HR") {ActualEffort} else if(ActualEffortUom == "DY") {ActualEffort*24} else if(ActualEffortUom == "WK") { ActualEffort*168} else if(ActualEffortUom == "MIN") {ActualEffort* (1/60)} else{null}

b. PlannedEffortHours

Groovy expression:

if(PlannedEffortUom=="MN"){PlannedEffort*720} else if(PlannedEffortUom == "HR") {PlannedEffort} else if(PlannedEffortUom == "DY") {PlannedEffort*24} else if(PlannedEffortUom == "WK") {PlannedEffort*168} else if(PlannedEffortUom == "MIN") {PlannedEffort* (1/60)} else{null}

c. EffortDifference

Groovy expression:

if (ActualEffortHours == null || PlannedEffortHours == null) {null} else { ActualEffortHours – PlannedEffortHours}

d. TotalHoursPlanned

Groovy expression:

adf.object.SrtasksTbl.sum("(PlannedEffortHours == null) ? 100000: PlannedEffortHours")

e. TotalHoursActual

Groovy expression:

adf.object.SrtasksTbl.sum(“(ActualEffortHours == null) ? 100000: ActualEffortHours”)

f. ***TotalHoursPlannedFinal***

Groovy expression:

if (TotalHoursPlanned >= 100000) {null}else{TotalHoursPlanned}

g. ***TotalHoursActualFinal***

Groovy expression:

if (TotalHoursActual >= 100000) {null}else{TotalHoursActual}

h. ***NumberofTasks***

Groovy expression:

adf.object.SrtasksTbl.count(“TaskNumber”)

3. View Object and Page Display

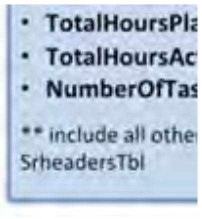


Figure 50. Maintenance Effort Report (Bravo) View Objects.

Figure 51. Maintenance Effort Report (Bravo)Tab Design.

C. SECONDARY REPARABLE REPORT (CHARLIE)

1. View Objects

a. CNsnDisplayVO

SELECT XxmcR001ItemmasterTbl.TAMCN,
XxmcR001ItemmasterTbl.RECORD_NSN,
inventory.ACTIVITY_ADDRESS_CODE,
XxmcR001ItemmasterTbl.NOMENCLATURE,
SUM(inventory.QUANTITY_ONHAND) as QOH FROM
XXMC_R001_ITEMMASTER_TBLXxmcR001ItemmasterTbl,XXMC_R001
_INVENTORY_TBLinventory WHERE
XxmcR001ItemmasterTbl.RECORD_NSN = inventory.RECORD_NSN and
XxmcR001ItemmasterTbl.FLOAT_IND='F' and (inventory.RECORD_NSN,
inventory.ACTIVITY_ADDRESS_CODE)in
(SELECTheaders.NSN_IN_MAINTENANCE,
headers.OWNER_UNIT_ADDRESS_CODE
FROM XXMC_R001_SRHEADERS_TBL headers
WHERE headers.MILITARY_LABOR_HOURS is not null)

GROUPBYXxmcR001ItemmasterTbl.RECORD_NSN,
inventory.ACTIVITY_ADDRESS_CODE,
XxmcR001ItemmasterTbl.TAMCN,
XxmcR001ItemmasterTbl.NOMENCLATURE

b. CHeadersDisplayVO

SELECT *FROM XXMC_R001_SRHEADERS_TBL SrheadersTblEO
WHERE SrheadersTblEO.NSN_IN_MAINTENANCE in
(SELECTUNIQUE(XxmcR001ItemmasterTbl.RECORD_NSN)
FROM XXMC_R001_ITEMMASTER_TBL XxmcR001ItemmasterTbl,
XXMC_R001_INVENTORY_TBL inventory
WHEREXxmcR001ItemmasterTbl.RECORD_NSN
inventory.RECORD_NSN and
XxmcR001ItemmasterTbl.FLOAT_IND = 'F'
GROUPBYXxmcR001ItemmasterTbl.RECORD_NSN,
inventory.ACTIVITY_ADDRESS_CODE,
XxmcR001ItemmasterTbl.TAMCN,
XxmcR001ItemmasterTbl.NOMENCLATURE)

2. Transient Attributes

a. NumbersOfSR

Groovy expression:
adf.object.CHeadersDisplay.count("SrNumber")

b. TotalManHours

Groovy expression:
adf.object.CHeadersDisplay.sum("MilitaryLaborHours")

c. NumSRNotNull

Groovy expression:
adf.object.CHeadersDisplay.count("MilitaryLaborHours")

d. PercentMissing

Groovy expression:

if (NumberOfSR == null){null}else{(1 – (NumSRNotNull / NumberOfSR)) *100}

3. View Object and Page Design

Figure 52. Secondary Reparable Report (Charlie) View Objects.

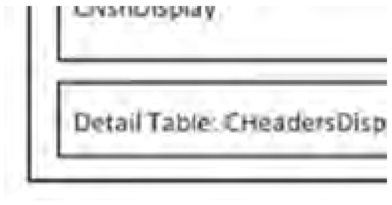


Figure 53. Secondary Reparable Report (Charlie) Tab Design.

D. COST REPORT (DELTA)

1. View Objects

a. DheadersviewVO

SELECT *FROM XXMC_R001_SRHEADERS_TBL h

```
WHERE h.sr_number in(SELECT head.sr_number
FROMXXMC_R001_SRHEADERS_TBLhead,XXMC_R001_REPAIRPAR
TS_TBLr,XXMC_R001_ITEMMASTER_TBLitemWHEREhead.sr_number
=r.sr_numberandr.parts_charge>0andr.quantity_required>=1and
r.record_nsn= item.record_nsnand item.stores_account_cd = 1 and
r.document_numberin(SELECTd.document_numberFROMXXMC_R001_
DUEIN_TBLdWHEREd.document_identifier_code='A0A'or
d.document_identifier_code=
'A01'unionallSELECTh.document_numberFROMXXMC_R001_HIST_DUE
IN_TBL hWHERE h.document_identifier_code= 'A0A' or
h.document_identifier_code= 'A01'))
```

b. RepairpartsTblVO

```
SELECT RepairpartsTblEO.SR_NUMBER,
RepairpartsTblEO.TASK_NUMBER,
RepairpartsTblEO.SERVICE_ACTIVITY,
RepairpartsTblEO.ORG_CODE,
RepairpartsTblEO.RECORD_NSN,
RepairpartsTblEO.QUANTITY_REQUIRED,
RepairpartsTblEO.PARTS_CHARGE,
RepairpartsTblEO.DOCUMENT_NUMBER,
RepairpartsTblEO.STATUS_DATE,
RepairpartsTblEO.DATE_RECEIVED_CANCELLED,
RepairpartsTblEO.DEMAND_CODE,
RepairpartsTblEO.SUPP_STATUS_DIC,
RepairpartsTblEO.SUPPLY_STATUS_CODE,
RepairpartsTblEO.SIGNAL_CODE,
RepairpartsTblEO.FLIGHT_STATUS,
RepairpartsTblEO.ERROR_MESSAGE,
RepairpartsTblEO.PROCESS_STATUS,
RepairpartsTblEO.RECORD_ID,
RepairpartsTblEO.CREATED_BY,
RepairpartsTblEO.CREATION_DATE,
RepairpartsTblEO.LAST_UPDATED_BY,
RepairpartsTblEO.LAST_UPDATE_DATE,
RepairpartsTblEO.REQUEST_ID,
RepairpartsTblEO.BATCH_ID,
RepairpartsTblEO.EXTERNAL_APPLICATION,
itemmaster.NOMENCLATURE,
itemmaster.STORES_ACCOUNT_CD
FROM XXMC_R001_REPAIRPARTS_TBLRepairpartsTblEO,
XXMC_R001_ITEMMASTER_TBL itemmaster
WHERERepairpartsTblEO.PARTS_CHARGE>0AND
RepairpartsTblEO.QUANTITY_REQUIRED>=1and
itemmaster.RECORD_NSN=RepairpartsTblEO.RECORD_NSN and
itemmaster.STORES_ACCOUNT_CD=1and
RepairpartsTblEo.DOCUMENT_NUMBER in
(SELECT d.document_numberFROM XXMC_R001_DUEIN_TBL d
WHEREd.document_identifier_code='A0A'or d.document_identifier_code=
'A01'unionallSELECTh.document_numberFROMXXMC_R001_HIST_DUE
IN_TBLhWHEREh.document_identifier_code='A0A'or
h.document_identifier_code= 'A01')
```

c. DRACTotalsVO

```
SELECT Dheadersview.REGIONAL_ACTIVITY_CODE,
sum(r.PARTS_CHARGE * r.QUANTITY_REQUIRED) as TotalCost
FROMDHEADERSVIEWDheadersview,XXMC_R001_REPAIRPARTS_TB
Lr ,XXMC_R001_ITEMMASTER_TBLitemmasterWHEREr.PARTS_CHAR
GE>0and r.QUANTITY_REQUIRED >= 1 and itemmaster.RECORD_NSN
= r.RECORD_NSN and itemmaster.STORES_ACCOUNT_CD = 1 and
Dheadersview.SR_NUMBER=r.SR_NUMBERandDheadersview.REGION
AL_ACTIVITY_CODE is not null and r.DOCUMENT_NUMBER
```

in(SELECTd.document_numberFROMXXMC_R001_DUEIN_TBLdWHERE d.document_identifier_code = 'A0A' or d.document_identifier_code= 'A01'unionallSELECTh.document_numberFROMXXMC_R001_HIST_DUEIN_TBLhWHEREh.document_identifier_code='A0A'or h.document_identifier_code='A01')GROUPBY Dheadersview.REGIONAL_ACTIVITY_CODE

d. DOUACTotalsVO

SELECT Dheadersview.OWNER_UNIT_ADDRESS_CODE, Dheadersview.REGIONAL_ACTIVITY_CODE,sum(r.PARTS_CHARGE*r.QUANTITY_REQUIRED)asUnitTotalCost FROMDHEADERSVIEW Dheadersview,XXMC_R001_REPAIRPARTS_TBLr,XXMC_R001_ITEMMASTER_TBLitemmasterWHEREr.PARTS_CHARGE>0andr.QUANTITY_REQUIRED>=1anditemmaster.RECORD_NSN=r.RECORD_NSNanditemmaster.STORES_ACCOUNT_CD=1 and Dheadersview.SR_NUMBER = r.SR_NUMBERandDheadersview.REGIONAL_ACTIVITY_CODEisnotnull andDheadersview.OWNER_UNIT_ADDRESS_CODE is not null and r.DOCUMENT_NUMBERin(SELECTd.document_numberFROM XXMC_R001_DUEIN_TBL dWHERE d.document_identifier_code = 'A0A' ord.document_identifier_code='A01'unionallSELECTh.document_number FROMXXMC_R001_HIST_DUEIN_TBLhWHEREh.document_identifier_code='A0A'orh.document_identifier_code='A01')GROUPBYDheadersview.OWNER_UNIT_ADDRESS_CODE,Dheadersview.REGIONAL_ACTIVITY_CODE

e. DPieChartVO

SELECT RepairpartsTblEO.RECORD_NSN, RepairpartsEO.SR_NUMBER,SUM(RepairpartsTblEO.QUANTITY_REQUIRED*RepairpartsTblEO.PARTS_CHARGE)asPartsCostFROMXXMC_R001_REPAIRPARTS_TBLRepairpartsTblEO,XXMC_R001_ITEMMASTER_TBLitemWHERERepairpartsTblEO.PARTS_CHARGE>0and RepairpartsTblEO.QUANTITY_REQUIRED>=1and

RepairpartsTblEO.RECORD_NSN=item.RECORD_NSNand
item.STORES_ACCOUNT_CD=1and
RepairpartsTblEO.DOCUMENT_NUMBER in
(SELECT d.document_numberFROM XXMC_R001_DUEIN_TBL d
WHEREd.document_identifier_code='A0A'or d.document_identifier_code=
'A01'unionallSELECTh.document_numberFROM
XXMC_R001_HIST_DUEIN_TBL hWHERE h.document_identifier_code=
'A0A'orh.document_identifier_code='A01')GROUPBY
RepairpartsTblEO.RECORD_NSN, RepairpartsTblEO.SR_NUMBER

2. Transient Attributes

a. PartsCost

Groovy expression:
PartsCharge*QuantityRequired

b. TotalCost

Groovy expression:
adf.object.RepairpartsTbl.sum(“PartsCost”)

c. NumberOfAssociatedRepairParts

Groovy expression:
adf.object.RepairpartsTbl.sum(“QuantityRequired”)

3. View Objects and Page Design

Figure 54. Cost Report (Delta) View Objects 1.

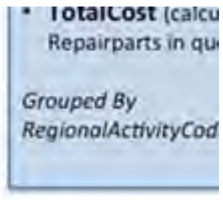


Figure 55. Cost Report (Delta) View Objects 2.

E. GMRT MAINTENANCE PRODUCTION REPORT (ECHO)

1. View Objects

a. EHeadersDisplayVO

```
SELECT*FROM XXMC_R001_SRHEADERS_TBLSrheadersTblEO
WHERESrheadersTblEO.SR_NUMBERin(SELECTt.INCIDENT_NUMBER
FROM XXMC_R001_SRTASKS_TBL t   WHERE t.TASK_TYPE = 'Supply'
AND(t.TASK_STATUS != 'Completed'   AND t.TASK_STATUS !=
'Approved' AND t.TASK_STATUS != 'Closed' AND t.TASK_STATUS !=
'Cancelled' AND t.TASK_STATUS != 'Rejected'))
```

b. ETasksVO

```
SELECT *FROM  XXMC_R001_SRTASKS_TBL SrtasksTblEO
WHERE(SrtasksTblEO.TASK_STATUS!='Completed'AND
SrtasksTblEO.TASK_STATUS!='Approved'AND
SrtasksTblEO.TASK_STATUS!='Closed'AND
SrtasksTblEO.TASK_STATUS!='Cancelled'AND
SrtasksTblEO.TASK_STATUS != 'Rejected')
```

c. DueinTblVO

```
SELECT *FROM  XXMC_R001_DUEIN_TBL DueinTblEO
```

d. DueinStatTblVO

```
SELECT *FROM  XXMC_R001_DUEIN_STAT_TBL DueinStatTblEO
```

2. Transient Attributes

a. Number of Tasks

Groovy expression:

```
adf.object.ETasks.count(“TaskNumber”)
```

b. Number of Orders

Groovy expression:

```
adf.object.DueinTbl.count(“RecordID”)
```

3. View Objects and Page Design

Figure 56. GMRT Maintenance Production Report (Echo) View Objects.

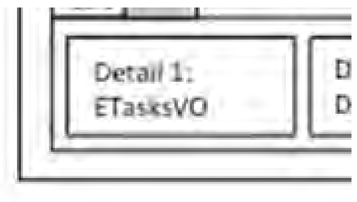


Figure 57. GMRT Maintenance Production Report (Echo) Tab Design 1.

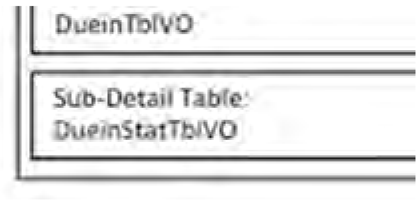


Figure 58. GMRT Maintenance Production Report (Echo) Tab Design 2.

F. DOCUMENT NUMBER REPORT (FOXTROT)

1. View Objects

a. FheadersviewVO

SELECT*FROMXXMC_R001_SRHEADERS_TBLWHERE
Sr_numberin(SELECTsr_numberFROMXXMC_R001_DUEIN_TBLGROU
PBYsr_numberunionSELECTsr_numberFROMXXMC_R001_HIST_DUEI
N_BL GROUP BY sr_number)

b. FDueinAll

SELECT 1 as SOURCE_TABLE
ADVICE_CODE,
APPROVED_DATE,
BACK_ORDERED_QTY,
BATCH_ID,
CANCELLED_QTY,
CONDITION_CODE,
CREATED_BY,
CREATION_DATE,
DEMAND_CODE,
DESTINATION_SUBINVENTORY,
DISTRIBUTION_CODE,
DOCUMENT_IDENTIFIER_CODE,
DOCUMENT_NUMBER,
ERROR_MESSAGE,

```
FLIGHT_STATUS,
JOB_ORDER_NUMBER,
LAST_STATEMENT_DATE,
LAST_UPDATE_DATE,
LAST_UPDATED_BY,
NEXT_FOLLOW_UP,
ORDER_NUMBER,
ORDERED_QTY,
PRIORITY_CODE,
PROCESS_STATUS,
PROJECT_CODE,
PURPOSE_CODE,
RAC,
RECORD_ID,
RECORD_NSN,
REQUEST_ID,
REQUIRED_DELIVERY_DATE,
RIC,
SHIP_FROM_ORG,
SHIP_RECEIVED_QTY,
SHIP_TO_ORG,
SIGNAL_CODE,
SR_NUMBER,
SUPPLEMENTAL_ADDRESS,
TASK_NUMBER,
TRANSACTION_TYPE,
UNIT_OF_ISSUE,
UNIT_PRICE
FROM DueinTblEO
UNION ALL
```

```
SELECT 2 as SOURCE_TABLE
ADVICE_CODE,
APPROVED_DATE,
BACK_ORDERED_QTY,
BATCH_ID,
CANCELLED_QTY,
CONDITION_CODE,
CREATED_BY,
CREATION_DATE,
DEMAND_CODE,
DESTINATION_SUBINVENTORY,
DISTRIBUTION_CODE,
DOCUMENT_IDENTIFIER_CODE,
DOCUMENT_NUMBER,
ERROR_MESSAGE,
FLIGHT_STATUS,
JOB_ORDER_NUMBER,
LAST_STATEMENT_DATE,
LAST_UPDATE_DATE,
LAST_UPDATED_BY,
NEXT_FOLLOW_UP,
ORDER_NUMBER,
ORDERED_QTY,
PRIORITY_CODE,
PROCESS_STATUS,
PROJECT_CODE,
PURPOSE_CODE,
RAC,
RECORD_ID,
RECORD_NSN,
REQUEST_ID,
REQUIRED_DELIVERY_DATE,
RIC,
SHIP_FROM_ORG,
SHIP_RECEIVED_QTY,
SHIP_TO_ORG,
SIGNAL_CODE,
SR_NUMBER,
SUPPLEMENTAL_ADDRESS,
TASK_NUMBER,
TRANSACTION_TYPE,
UNIT_OF_ISSUE,
UNIT_PRICE
FROM HistDueinTblEO
```

2. Transient Attributes

a. Received

Groovy expression:

```
 if (SourceTable == 1) {“No”} else {“Yes”}
```

b. Number of Associated Documents

Groovy expression:

```
adf.object.FDueinAll.sum(“(SourceTable == 2) ? 1: SourceTable”)
```

c. Number of Closed Documents

Groovy expression:

```
(adf.object.FDueinAll.sum(“(SourceTable == 1) ? 0: SourceTable”)) / 2
```

d. Percent of Documents Closed

Groovy expression:

```
(NumberOfDocumentsClosed / NumberOfAssociatedDocuments) * 100
```

3. View Objects and Page Design

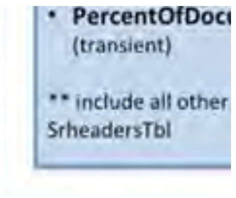


Figure 59. Document Number Report (Foxtrot) View Objects.

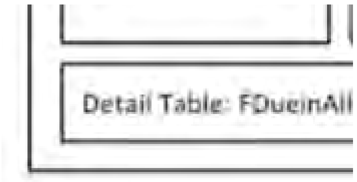


Figure 60. Document Number Report (Foxtrot) Tab Design 1.

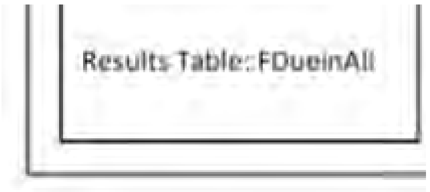


Figure 61. Document Number Report (Foxtrot) Tab Design 2.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. IMPLEMENTATION STEPS

Figures 62 through 83 demonstrate screen shots of the implementation steps for all six Use Cases.

A. GENERAL STEPS

1. Create a connection in JDeveloper to a clean copy of the database.
2. Use the Create Business Components from Tables' wizard to create a default Entity Object and View Object in JDeveloper based off of each of the relevant database tables.
3. For each report:
    a. Create relevant (read-only) View Objects for the report using one of the following methods:
        i. Edit a default View Object if that database table will only be used once in the application.
        ii. Create a new Entity-Object-based View Object if the SQL is based on one or two specific tables and only requires adding a WHERE clause. If the SQL is more complex, one may be able to create an Entity-Object-based View Object and then edit the Query in Expert Mode.
        iii. Create a Query-based View Object using the Create Business Components from Tables' wizard if the View Object will *not* retain the primary key from at least one of the database tables that it is based off of (i.e., if the object will be created using a Group By statement).
        iv. Implement the SQL statement in SQL Developer if it is very complex. Then go through the Create Business Components from Tables' wizard to create an Entity Object and a default View Object based off of this database view.

b. Add view links between each of the View Objects for the report. Usually these will connect one attribute in each View Object; however, there are cases where a link between two objects will connect two or three attributes in each.
c. Create the appropriate transient attributes in each view object. If a transient attribute is accessing data in another View Object (i.e., for a Master-Detail relationship), make sure to utilize the correct View Accessor name (found in the view link).
d. For any tables that will be linked directly to a Query Panel (usually the Master table for the report), create a View Criteria with items for each of the attributes that should be searchable. Each View Criteria item should be set to equal a Literal, with the value field of that Literal left blank.
e. Set the Tuning for each View Object under the General tab of the editor, and set the UI Hints (especially the Label) for each attribute under the Attributes tab.
    i. The Range Size in the Tuning section of the View Object should be equal to the Range Size that is set for the table(s) based off of that View Object. The field that says “in batches of” in the Tuning section should be set to Range Size + 3.
f. Create an application module for the Use Case and add all relevant View Objects. If one table is a Detail of another, it should be added underneath the Master table. If multiple Detail tables will be shown under one Master table, these should all be added under the *same* instance of the Master table.
g. Create a new page for the report.
    i. If the report will display multiple levels of data and requires a Query Panel, the Query Panel should be added separately (rather than as an ADF Query Panel with Table).

        1. *Note: For all Query Panels in this application, the following settings were changed in the Property Inspector: ModeChangeVisible = False and SaveQueryMode = Hidden.*

    ii. Once the Query Panel is added, the Master-Detail table should be put in separately by dragging the Detail table onto the page.

        1. The ID field of the Master table should then be entered into the Query Panel Property Inspector in the ResultComponentID field.
        2. For all tables, one column should be set in the Property Inspector so that RowHeader = unstyled.
        3. All tables should also have a value in the Summary field of the Property Inspector to avoid warnings in JDeveloper.

    iii. Additional Detail or Sub-detail (under the Detail table) tables can be added to the page by dragging the instances of these tables that are under the *same instance of the Master table* that is already on the page. By doing this, the tables will automatically sync up with the selected row in the Master table.

    iv. Adjust the page layout with different ADF Faces components.

    v. Add charts as desired. All charts and gauges must have a value in the Summary field in the Property Inspector.

4. Create a main page and implement navigation between this and the various report pages.

B. READINESS REPORT (ALPHA)

Page Layout

The “Main Report” ShowDetailItem for this report is structured with two vertical PanelGroupLayouts inside of a main PanelGroupLayout (PGL) that is set to scroll. The first PGL holds the Query Panel, while the second holds three PanelHeaders (separated by spacers) with each of the relevant report tables.

The “Deadlined Items” ShowDetailItem also contains a PanelGroupLayout set to scroll. Inside this is a single vertical PGL that contains one PanelHeader (holding the Query Panel) and a table.

Figure 62. Top-level structure for the Alpha page.

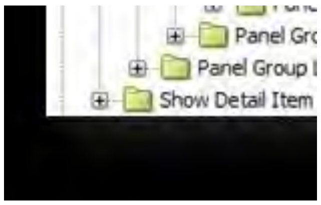


Figure 63. Fully-expanded structure for the “Main Report” tab of the Alpha page.

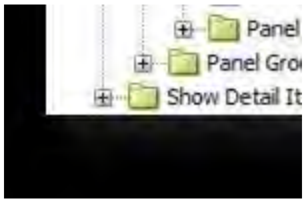
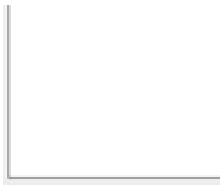
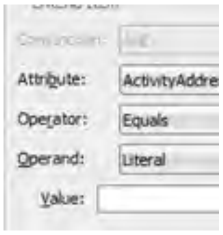


Figure 64. Fully-expanded structure for the “Deadlined Items” tab of the Alpha page.

Basic Steps

1. Create all view objects, view links, and transient attributes (note: the transient attributes related to TotalDeadlined should all be based on AHeadersUniqueVO rather than AHeadersDisplayVO).
    a. Set labels in the UI Hints tab for each of the three view objects that will be displayed to the user (ATamClassVO, ATamcnDisplayVO, AHeadersDisplayVO).
    b. In the General tab, adjust the tuning for each of these.
        i. ATamClassVO:
            1. Range Size = 5
            2. in batches of = 8
        ii. ATamcnDisplayVO:
            1. Range Size = 10
            2. in batches of = 13
        iii. ANsnDisplay:
            1. Range Size = 20
            2. in batches of = 23
2. Create two view criteria.
    a. In ATamClassVO:





Conjunction: AND

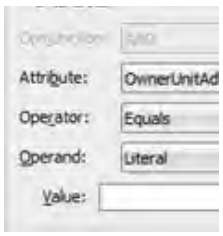
Attribute: ActivityAddre

Operator: Equals

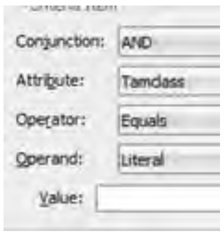
Operand: Literal

Value:

b. In AHeadersDisplayVO:

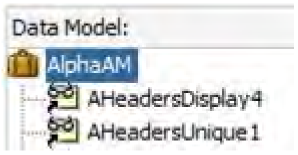


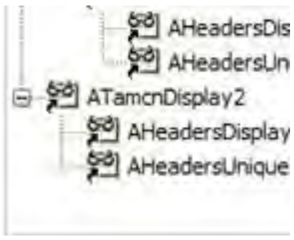
Conjunction:	AND
Attribute:	OwnerUnitAd
Operator:	Equals
Operand:	Literal
Value:	



Conjunction:	AND
Attribute:	Tamclass
Operator:	Equals
Operand:	Literal
Value:	

3. Create a new application module for Alpha as shown next:





4. Create a new page based on the XXMC Reports template and enter the useCaseTitle.
    a. Construct the page’s toolbar.
5. Add a PanelTabbed component to the mainContent window and name the ShowDetailItem “Main Report.”
6. Add a PanelGroupLayout within the main tab and set it to scroll.
    a. Add a spacer within this.
7. Drag the named criteria ATamClassVOCriteria (located under ATamClass in the Data Control) onto the page below the spacer as a Query Panel.
    a. Change the text of the PanelHeader for this Query Panel and adjust the Query Panel settings as described in the “General Steps” section of this document.
8. Drag the instance of ATamcnDisplay under ATamClass onto the page below the PanelGroupLayout for the Query Panel as a Master-Detail table.
    a. Copy the ID of the Master table and paste it into the ResultComponentID field in the properties of the Query Panel.
    b. Change the text of the PanelHeaders for both tables.
    c. Re-order the columns for both tables and enable sorting and filtering on the Detail table.
9. Drag the instance of AHeadersDisplay under ATamcnDisplay under ATamClass into the PanelGroupLayout containing the other two tables.

    a. Choose the relevant attributes.
    b. Surround this table with a PanelHeader by right-clicking it in the Structure Pane and choosing the “Surround With” option.

10. Add spacers between each of the PanelHeaders within the PanelGroupLayout that holds the tables.
11. Set the range sizes of the tables to:
    a. Master – 5
    b. Detail – 10
    c. Sub-detail – 20
12. Add a Gauge column to the Master table.
    a. Open the editor (click the pencil button) for the Master table and add a new column, based on the ReadinessPercent attribute. There will now be two columns showing ReadinessPercent.
    b. Exit the editor and drag a Gauge component from the Component Palette onto the second Readiness Percent column.
        i. Choose an LED style gauge with no title or key.
    c. Right-click the outputText attributes under the second Readiness Percent column (where the gauge now is) in the Structure Pane and delete it.
    d. Configure the gauge:
        i. Set the Value field in the Property Inspector to #{row. ReadinessPercent} using the Expression Builder.
            1. Under the Gauge Data section in the Property Inspector, put in a MinValue of 0 and a MaxValue of 100.
            2. Use the InlineStyle field to change the size of the gauge: “width:35px;height:35px”
        ii. Go to the Metric Label component under the Gauge in the Structure Pane and for Position select LP_NONE.

        iii. Expand the dvt:thresholdSet component under the Gauge in the Structure Pane.
            1. In the first dvt:threshold component, set the FillColor to red (#d90000) and the ThresholdMaxValue to 85.0.
            2. In the second, set the FillColor to golden yellow (#ffd642) and the ThresholdMaxValue to 94.0.
            3. In the third, set the FillColor to green (#00a500) and the ThresholdMaxValue to 100.0.
        iv. Click on the column with the gauge in it in the Structure Pane, and under the Appearance tab of the Property Inspector, set the Align property to "Center."
            1. Set the column Width to 75.
                a. Also change the width of the columns Total On Hand, Number Deadlined and Readiness Percent.
            2. Set the column HeaderText to "ReadinessMeter."

13. Repeat the above steps to insert gauge components into the Tamcn table (1st detail level). The only difference will be the setting in the InlineStyle field of the Gauge component, where height and width should be 25 instead of 35.
14. Add a new ShowDetailItem to the PanelTabbed component of the page and title it "Deadlined Items."
    a. Add a Panel Group Layout with layout set to Scroll.
    b. Drag the named view criteria AHeadersDisplayVOCriteria under AHeadersDisplay into the Panel Group Layout as a Query Panel with Table.
        i. Enable sorting, filtering, and single row selection in the table and choose which attributes will be displayed.
    c. Change the text of the Panel Header and the settings of the Query Panel.

C. MAINTENANCE EFFORT REPORT (BRAVO) (VERSION 1 – CONTAINS NULL VALUES)

Page Layout

The “Main Report” ShowDetailItem for this report is structured with a vertical PanelSplitter inside of a main PanelGroupLayout (PGL) that is set to scroll. The 1st facet of the PanelSplitter contains a scroll PanelGroupLayout surrounding a spacer and two vertical PGLs. The first vertical PGL is for the PanelHeader with the page’s Query Panel. The second holds two horizontal PanelSplitters separated by a spacer.

1. The 1st facet of the upper PanelSplitter contains a PanelHeader holding the Master table for the report. The 2nd facet contains a PanelFormLayout that shows a detailed view of the information in the Master table. This PanelFormLayout is situated inside of a PanelGroupLayout, which is inside of a PanelHeader, which is inside another PanelGroupLayout *(note: This nested structure of Panel Group Layouts was originally created to allow spacers to be inserted in order to achieve the desired positioning of the PanelFormLayout on the page. We later discovered that the Padding attribute of the PanelHeader could be used to achieve the same effect – therefore, the structure of Version 2 of this page is simpler and more efficient and should be used in place of this structure).*
2. The 1st facet of the lower PanelSplitter contains a PanelHeader holding the Detail table for the report. The 2nd facet contains a PanelGroupLayout holding a Gauge component and an OutputText component (which displays the value of the gauge).

The 2nd facet of the outermost vertical PanelSplitter contains a horizontal PanelGroupLayout holding the graph components for the page. Inside this PanelGroupLayout are two PanelHeaders, the first holding a PieGraph and the second holding a BarGraph.

Figure 65. Top-level structure for the Bravo page (with nulls).

Figure 66. Detailed structure of the “Main Report” for the Bravo page.

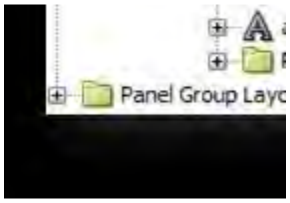


Figure 67. Structure of horizontal Panel Splitters for the Bravo page.

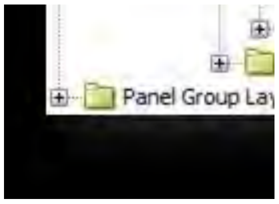


Figure 68. Structure of PanelGroupLayout holding the graph components for the Bravo page.

Basic Steps

1. Create all view objects.
    a. SrtasksTbl is simply the default view object created by JDeveloper from the SrtasksTbl entity object. BHeadersDisplayVO is a new object but can easily be created using a wizard; it is based on SrheadersTblEO and has a WHERE clause which is shown in the "Overview of Pages" documentation.
2. In SrtasksTblEO (entity object), create the two transient attributes listed in the "Overview of Pages" documentation.
    a. In the Attribute page of SrtasksTblVO (view object), click the plus button to "Add Attribute from Entity" and add in the two transient attributes that were just created in SrtasksTblEO.
3. Create all remaining view links and view-object-level transient attributes.
    a. Set labels in the UI Hints tab for both view objects.
        i. In BHeadersDisplayVO, set TotalHoursActual and TotalHoursPlanned to "Hide" instead of "Display." These

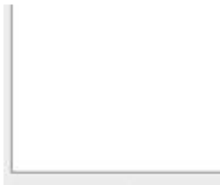
attributes are only used as part of an intermediate calculation.

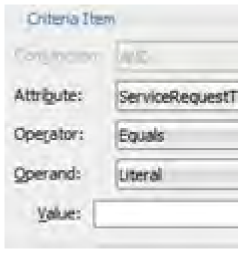
   b. In the General tab, adjust the tuning for each of these.
      i. BHeadersDisplayVO:
         1. Range Size = 20
         2. in batches of = 23
      ii. SrtasksTblVO:
         1. Range Size = 10
         2. in batches of = 13

4. Create a read-only view object named BSrHeadersTblVVO using the Create Business Components from Tables wizard.
   a. Go to the wizard page that says “Query-Based View Object” and create one based on SrheadersTbl.
   b. Exit the wizard and go to the Attributes tab of the view object.
      i. Delete all attributes except ServiceRequestType.
   c. Go to the Query tab and delete all of the other attributes from the SQL as well.
      i. Add a Group By statement to the query to group based on ServiceRequestType.
      ii. Add a WHERE clause to the SQL to get only the ServiceRequestTypes that are relevant to this Use Case (Maintenance – CAL, Maintenance – MOD, Maintenance – PM, Maintenance – CM, Maintenance – MISC, Maintenance – SL3, and Maintenance – SRP).
5. Create a read-only view object named MPCListVVO using the Create Business Components from Tables wizard.
   a. Go to the wizard page that says “Query-Based View Object” and create one based on SrheadersTbl.
   b. Exit the wizard and go to the Attributes tab of the view object.
      i. Delete all attributes except MasterPriorityCode.

    c. Go to the Query tab and delete all of the other attributes from the query as well, and add a statement to Group By MasterPriorityCode.

6. Under the Attribute tab of BHeadersDisplayVO, select the ServiceRequestType attribute.
    a. Go to the LOV tab for this attribute and click to add a new LOV.
        i. Within the wizard, click to add a new view accessor which points to the ServiceRequestType attribute in BSrHeadersTblVVO. The ServiceRequestType field should now show up in forms as a drop-down menu with options.
    b. Do the same for the MasterPriorityCode attribute to add an LOV based on the MasterPriorityCode attribute in MPCListVVO.
7. Create a view criteria in BHeadersDisplayVO:





Criteria Item

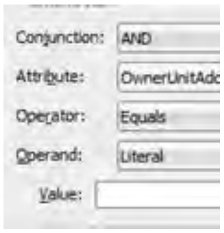
Conjunction: AND

Attribute: ServiceRequestT

Operator: Equals

Operand: Literal

Value:



Conjunction: AND

Attribute: OwnerUnitAdc

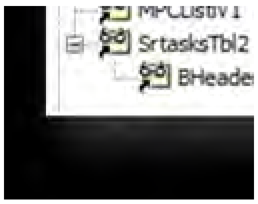
Operator: Equals

Operand: Literal

Value:

   a. In the UI Hints section of the View Criteria editor, click on the ServiceRequestType item and check the “Support Multiple Value Selection” box.

8. Create a new application module for Bravo (Version 1) as shown next:



9. Follow the steps listed below for Bravo Version 2 to create the page (the layout in this page is more efficient than the layout used in Version 1). Whenever the steps say to use “BHeadersViewVO,” instead use “BHeadersDisplayVO.”

D. MAINTENANCE EFFORT REPORT (BRAVO) (VERSION 2 – NO NULLS)

Page Layout

The “Main Report” ShowDetailItem for this report is structured with a vertical PanelSplitter inside of a main PanelGroupLayout (PGL) that is set to scroll. The 1st facet of the PanelSplitter contains a vertical PanelGroupLayout with a PanelHeader (for the Query Panel) and two horizontal PanelSplitters.

1. The 1st facet of the upper PanelSplitter contains a PanelHeader holding the Master table for the report. The 2nd facet contains a PanelFormLayout that shows a detailed view of the information in the Master table. This PanelFormLayout is situated inside of a PanelGroupLayout, which is inside of a PanelHeader.
2. The 1st facet of the lower PanelSplitter contains a PanelHeader holding the Detail table for the report. The 2nd facet contains a PanelGroupLayout holding a Gauge component and an OutputText component (which displays the value of the gauge).

The 2nd facet of the outermost vertical PanelSplitter contains a horizontal PanelGroupLayout holding the graph components for the page. Inside this PanelGroupLayout are two PanelHeaders, the first holding a PieGraph and the second holding a BarGraph.

Figure 69. Top-level structure for the Bravo page (no nulls).

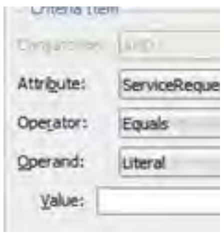
Figure 70. Structure of horizontal Panel Splitters for the Bravo page.

Figure 71. Structure of PanelGroupLayout holding the graph components for the Bravo page.

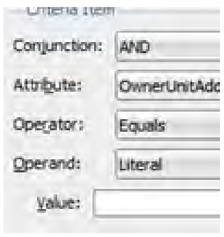
Basic Steps

1. Create a database view, BHeadersView, and add an entity object and a default view object in JDeveloper based off of this view. Edit the view object to add a WHERE clause, which is given in the “Overview of Pages” documentation.
2. Create all other view objects, view links, and transient attributes *(note: If Version 1 of Bravo has already been created, the SrtasksTblVO will already exist and does not need to be re-created. If Version 1 has not been created, SrtasksTblVO can be created by editing the default view object based off of SrtasksTblEO)*.
    a. Set labels in the UI Hints tab for both view objects.
    b. In the General tab, adjust the tuning for each of these.
        i. BHeadersViewVO:
            1. Range Size = 20
            2. in batches of = 23
        ii. SrtasksTblVO:
            1. Range Size = 10
            2. in batches of = 13
    c. Under the Attribute tab, select the ServiceRequestType attribute.
        i. Go to the LOV tab for this attribute and click to add a new LOV.
            1. Within the wizard, click to add a new view accessor which points to the ServiceRequestType attribute in BSrHeadersTblVVO. The ServiceRequestType field should now show up in forms as a drop-down menu with options *(note: There are steps on how to create BSrHeadersTblVVO in the documentation for Version 1 of Bravo).*
    d. Do the same for the MasterPriorityCode attribute to add an LOV based on the MasterPriorityCode attribute in MPCListVVO.

3. Create a view criteria in BHeadersViewVO:



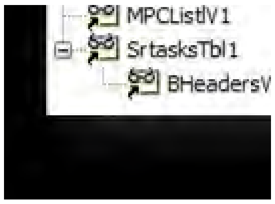
Criteria Item
Conjunction: AND
Attribute: ServiceReque
Operator: Equals
Operand: Literal
Value:



Criteria Item
Conjunction: AND
Attribute: OwnerUnitAdc
Operator: Equals
Operand: Literal
Value:

   a. In the UI Hints section of the View Criteria editor, click on the ServiceRequestType item and check the box to enable multiple selections.

4. Create a new application module for Bravo (Version 2) as shown next:



5. Create a new page based on the XXMC Reports template and enter the useCaseTitle.
    a. Construct the page’s toolbar.
6. Add a PanelTabbed component to the mainContent window and name the ShowDetailItem “Main Report.”
7. Add a PanelGroupLayout within the main tab and set it to scroll.
    a. Add a spacer within this.
8. Add a vertical PanelSplitter within the PanelGroupLayout.
    a. Set properties:
        i. PositionFromEnd: True
        ii. SplitterPosition: 350
        iii. Collapsed: False
9. Drag the Named Criteria BHeadersViewVOCriteria under BHeadersView into the 1st facet of the PanelSplitter as a Query Panel.
    a. Change the PanelHeader text and the Query Panel settings.
10. Add a Master-Detail table to the page under the query panel within the outer PanelGroupLayout by dragging the instance of SrTasks that is underneath SrHeadersView from the Data Control onto the page.
    a. Edit each table to choose the desired attributes to display.
        i. Turn off “Filterable” for the Service Request Type column in the Master table by selecting the column and using the Property Inspector.

        ii. Change the Range Size in the master table's property inspector to 20.

    b. Put the ID of the Master table into the ResultComponentID field of the Property Inspector for the Query Panel.
    c. Change the names of the PanelHeaders that enclose both tables.

11. Add a spacer between the two PanelHeaders that hold the Master and Detail tables (take the tables out of the PanelGroupLayout they come in).
12. Drag a PanelSplitter below the PanelHeader that holds the Query Panel and set layout to Horizontal.
    a. Drag the Master table into the 1st facet of the PanelSplitter.
    b. Drag the BHeadersView from the Data Control panel into the 2nd facet of the PanelSplitter and create a read-only ADF form.
    c. Set the Splitter Position in the Property Inspector.
13. Surround the PanelFormLayout in the 2nd facet of the PanelSplitter that holds the Master table with a PanelGroupLayout (by right-clicking the component in the Structure Pane) and set its layout to vertical.
    a. Put a spacer in the PanelGroupLayout before the PanelFormLayout.
    b. Surround the PanelGroupLayout with a PanelHeader and change the title.
        i. Set the PanelHeader to have Left Padding = 120px (approximately) in the Property Inspector.
14. Add graphics to the page.
    a. In the bottom panel of the outermost PanelSplitter for the page (which is vertical), drag the detail Srtasks table under the corresponding BHeadersView table as a Pie Chart.
        i. In the Pie Chart wizard, include the Actual Effort attribute as the "pie," and the Task Number and Task Name attributes as the "slices."
    b. Surround the pie chart with a PanelGroupLayout (horizontal).

        i. Drag the same detail Data Control based off of SrtasksTbl into this PanelGroupLayout to create a Bar Chart.
            1. In the Bar Chart wizard, add Actual Effort and Planned Effort for the bars. Add Task Number and Task Name for the x-axis.
    c. Surround each graphic with a PanelHeader to give them titles.
        i. Add a spacer in between the two graphs and adjust its width as necessary.

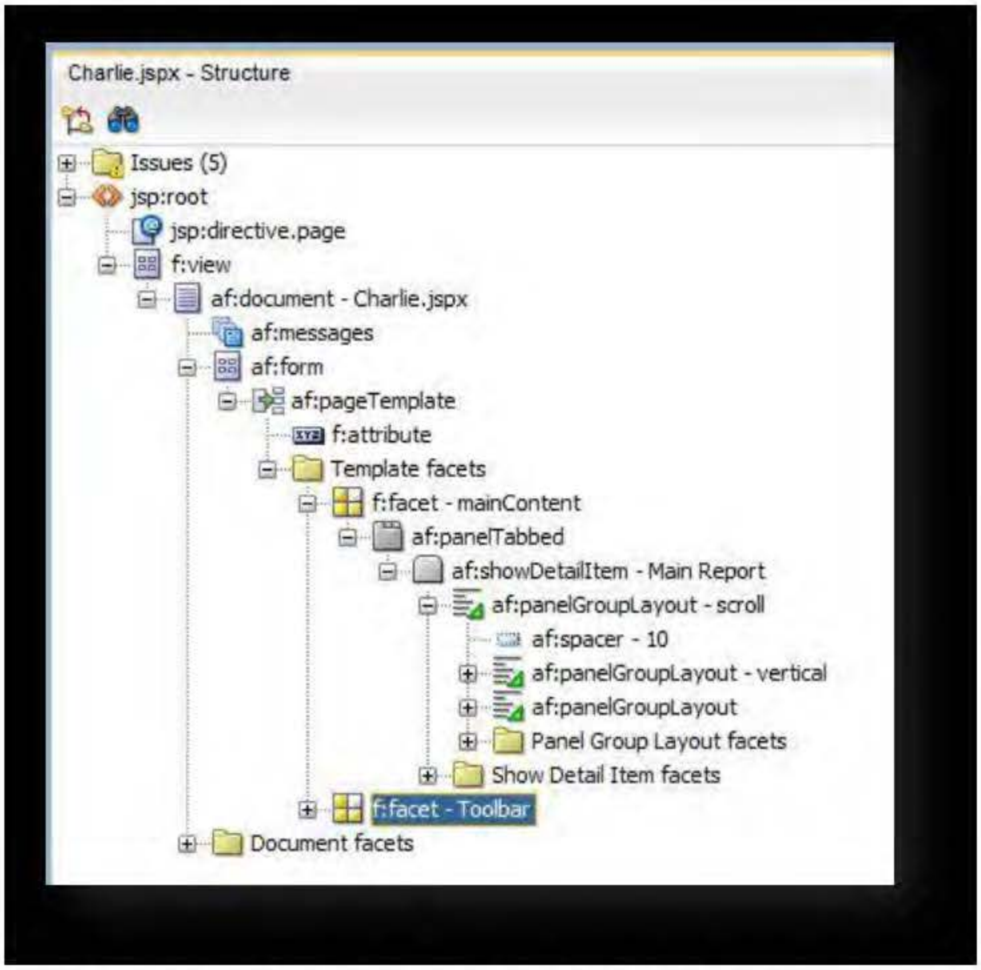
15. Add a gauge next to the Detail table on the page.
    a. Add a PanelSplitter to the Structure Pane above the Detail table but under the spacer below the Panel Splitter holding the Master table.
        i. Drag the Detail table into the 1st facet of this PanelSplitter.
        ii. Set the Splitter Position.
    b. Add a Gauge to the 2nd facet of the PanelSplitter by dragging the EffortDifference attribute of the SrTasks table that is underneath the BHeadersDisplay data control onto the page.
        i. Choose a half-circle style gauge component with no lower title and no legend.
            1. In the wizard, set the metric label to *not* show.
            2. Set the minimum value to -50 and the maximum to 50.
            3. Set thresholds so that the gauge is red from (-50,-25), yellow from (-25,-10), green from (-10,10), yellow from (10,25), and red from (25,50).
        ii. Set the padding on the left side of the gauge component to approximately 30 in the Property Inspector.
            1. Set the height and width of the gauge to 120 px using the InlineStyle field.
    c. Surround the Gauge with a horizontal PanelGroupLayout.

i. Insert an OutputText field that shows the value of the Gauge by dragging and dropping the EffortDifference attribute from the detail SrTasks Data Control into the PanelGroupLayout.
    1. Change the FontWeight to bold in the Property Inspector.

E. CHARLIE: SECONDARY REPARABLES REPORT

Page Layout

The "Main Report" ShowDetailItem for this report is structured with two vertical PanelGroupLayouts inside of a main PanelGroupLayout (PGL) that is set to scroll. The first vertical PGL holds a PanelHeader with a Query Panel, while the second holds two PanelHeaders (separated by a spacer) with each of the relevant report tables.



```
Issues (5)
jsp:root
    jsp:directive.page
    f:view
        af:document - Charlie.jspx
            af:messages
            af:form
                af:pageTemplate
                    f:attribute
                    Template facets
                        f:facet - mainContent
                            af:panelTabbed
                                af:showDetailItem - Main Report
                                    af:panelGroupLayout - scroll
                                        af:spacer - 10
                                        af:panelGroupLayout - vertical
                                        af:panelGroupLayout
                                        Panel Group Layout facets
                                    Show Detail Item facets
                        f:facet - Toolbar
            Document facets
```

Figure 72. Top-level structure for the Charlie page.

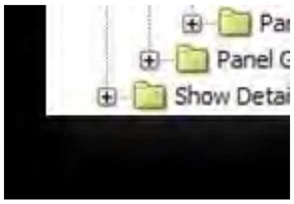
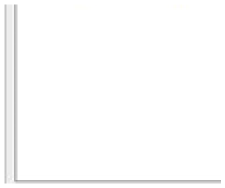
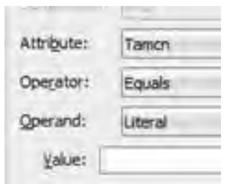


Figure 73. Detailed structure for the “Main Report” tab on the Charlie page.

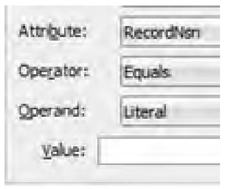
Basic Steps

1. Create all view objects, view links, and transient attributes.
    a. Set labels in the UI Hints tab for both of the view objects.
    b. In the General tab, adjust the tuning for each of these.
        i. CNsnDisplayVO:
            1. Range Size = 15
            2. in batches of = 18
        ii. CHeadersDisplayVO:
            1. Range Size = 15
            2. in batches of = 18
    c. Click on the RecordNsn and Tamcn attributes in the editor for CNsnDisplay and set their Width properties to 20 characters.
2. Create a view criteria in CNsnDisplayVO:

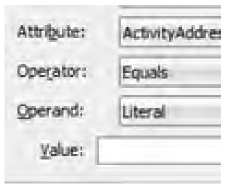




Attribute: Tamcn
Operator: Equals
Operand: Literal
Value:

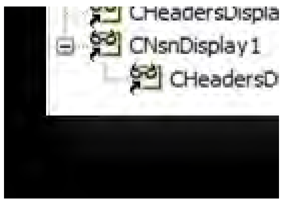


Attribute: RecordNsn
Operator: Equals
Operand: Literal
Value:



Attribute: ActivityAddre
Operator: Equals
Operand: Literal
Value:

3. Create a new application module for Charlie as shown next:



4. Create a new page based on the XXMC Reports template and enter the useCaseTitle.
    a. Construct the page's toolbar.
5. Add a PanelTabbed component to the mainContent window and name the ShowDetailItem "Main Report."
6. Add a PanelGroupLayout within the main tab and set it to scroll.
    a. Add a spacer within this.
7. Add a Query Panel to the page by dragging and dropping the Named Criteria CNsnDisplayVOCriteria (from the Data Control in a folder underneath CNsnDisplay) below the spacer in the PanelGroupLayout.
    a. Adjust the PanelHeader text and the Query Panel settings.
8. Add a Master-Detail table to the page by dragging the instance of CHeadersDisplay underneath CNsnDisplay and dropping it below the Query Panel PanelHeader.
    a. Change the text of the Panel Headers for the two tables.
    b. Put the ID of the Master table into the ResultComponentId field of the Query Panel in the Property Inspector.
    c. Select the relevant attributes to be included in each table by clicking the "Edit" (pencil) buttons in their Property Inspectors and enable sorting and single row selection.
    d. Add a spacer in between the Panel Headers for the Master and Detail tables.
9. Add a Gauge column to the Master table to indicate missing data.

a. In the Master table editor (reached by clicking the pencil button in the Property Inspector), add another column based on the PercentMissing attribute (there will now be two identical columns).
b. Drag a Gauge component from the Component Palette onto the column.
    i. From the menu, select a circular LED gauge with no title or key.
    ii. Delete the outputText attribute (in the Structure Pane) from under the column where the Gauge was just inserted.
c. Configure the Gauge:
    i. Set the Value field in the Property Inspector to #{row.PercentMissing} using the Expression Builder.
        1. Under Gauge Data, put in a MinValue of 0 and a MaxValue of 100.
        2. Use the InlineStyle field to change the size of the gauge: “width:15px;height:15px.”
    ii. Go to the Metric Label component under the Gauge in the Structure Pane and for Position select LP_NONE.
    iii. Expand the dvt:thresholdSet component under the Gauge in the Structure Pane and delete one of the dvt:threshold sub-components.
        1. In the first remaining dvt:threshold component, set the FillColor to white and the ThresholdMaxValue to 0.0.
        2. In the second, set the FillColor to orange (#ff8400) and the ThresholdMaxValue to 100.0.
    iv. Click on the Master table column with the Gauge in it, and under the Appearance tab of the Property Inspector, set Align to Center.
        1. Set the column Width to 75.

2. Set the column HeaderText to “Incomplete Data Warning.”

F. DELTA: COST REPORT

Page Layout

The “Main Report” ShowDetailItem for this report is structured with a vertical PanelGroupLayout and two horizontal PanelSplitters inside of a main PanelGroupLayout that is set to scroll. The inner PanelGroupLayout holds a PanelHeader with a Query Panel. Below this, the first PanelSplitter has a PanelHeader holding the Master table in its 1st facet. In its 2nd facet is a PanelHeader holding a PanelFormLayout with a detailed view of the data from the Master table. The second PanelSplitter has a PanelHeader holding the Detail table in its 1st facet and a PanelHeader holding a Pie Graph in its 2nd facet.

The “Unit Totals” ShowDetailItem also contains a PanelGroupLayout set to scroll. Inside this are two horizontal PanelGroupLayouts. The first horizontal PGL contains a PanelHeader with the page’s Master table, followed by a Pie Graph based on the same view object. Similarly, the second horizontal PGL contains a PanelHeader with the page’s Detail table, along with a Pie Graph.

Figure 74. Top-level structure for the Delta page.

Figure 75. Detailed structure of the “Main Report” tab of the Delta page.

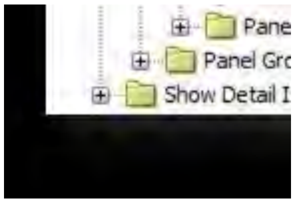


Figure 76. Detailed structure of the “Unit Totals” tab.

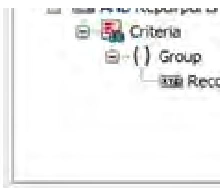
Basic Steps

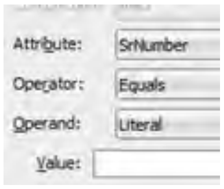
1. Create the Dheadersview view in the database using SQL Developer and then create a default entity and view object based on this view.
2. Edit RepairpartsTblVO (the default view object based off of RepairpartsTblEO) so that its query matches the one shown in the “Overview of Pages” documentation.
3. Create the remaining three view objects, along with all view links and transient attributes.
    a. Set labels in the UI Hints tab for all five view objects.
    b. In the General tab, adjust the tuning for each of these (except DPieChartVO, which is never displayed as a table).
        i. DheadersviewVO:
            1. Range Size = 15
            2. in batches of = 18
        ii. RepairpartsTblVO:
            1. Range Size = 15

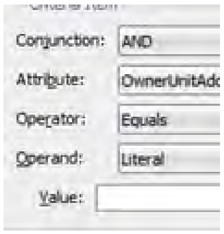
        2. in batches of = 18
    iii. DRACTotalsVO:
        1. Range Size = 10
        2. in batches of = 13
    iv. DOUACTotalsVO:
        1. Range Size = 20
        2. in batches of = 23

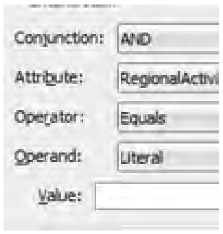
c. Click on the OwnerUnitAddressCode, RegionalActivityCode, and SrNumber attributes in the editor for DheadersviewVO and set their Width properties to 25. Do the same for RecordNsn in RepairpartsTblVO.

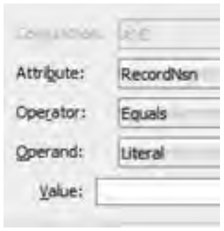
4. Create a view criteria in DheadersviewVO:




Attribute: SrNumber
Operator: Equals
Operand: Literal
Value:


Conjunction: AND
Attribute: OwnerUnitAdc
Operator: Equals
Operand: Literal
Value:


Conjunction: AND
Attribute: RegionalActivi
Operator: Equals
Operand: Literal
Value:


Attribute: RecordNsn
Operator: Equals
Operand: Literal
Value:

Note: The RecordNsn attribute must be accessed by first creating a new View Criteria Item based on "RepairpartsTbl." This will generate automatically generate a group with an Item inside it, and the inner item can then be set to any attribute from RepairpartsTblVO.

5. Create a new application module for Delta as shown below. Make sure that there is an instance of DPieChart and an instance of RepairpartsTbl under the *same* instance of Dheadersview. This is the instance of Dheadersview that should be used on the page.

6. Create a new page based on the XXMC Reports template and enter the useCaseTitle.
    a. Construct the page's toolbar.
7. Add a PanelTabbed component to the mainContent window and name the ShowDetailItem "Main Report."
8. Add a PanelGroupLayout within the main tab and set it to scroll.
    a. Add a spacer within this.
9. Add a Query Panel underneath the spacer by dragging and dropping the named view criteria that is under Dheadersview in the Data Control.
    a. Adjust the PanelHeader text and the settings of the Query Panel.
10. Add a Master-Detail table to the page under the Query Panel (within the outer PanelGroupLayout) by dragging the instance of RepairpartsTbl that is underneath Dheadersview.
    a. In both tables, include only the relevant attributes and enable sorting and filtering.
        i. Change the Range Size in the Master table's Property Inspector to 15.
    b. Enter the ID for the Master table into the ResultComponentID section of the Query Panel's Property Inspector.
    c. Change the names of the PanelHeaders that enclose both tables.

11. Drag a PanelSplitter into the outer PanelGroupLayout, just below PanelGroupLayout that holds the Query Panel.
    a. Set layout to Horizontal and adjust the Splitter Position.
    b. Drag the PanelHeader with the Master table into the 1st facet of the PanelSplitter.
12. Add a spacer below the PanelSplitter.
13. Drag Dheadersview from the Data Control panel into the 2nd facet of the PanelSplitter and create an ADF read-only form.
    a. Surround the PanelFormLayout with a PanelHeader (right click and choose "surround With") and change the PanelHeader's title.
    b. Set the Padding Left for the PanelHeader so that there is sufficient space between it and the Master table.
14. Add another horizontal PanelSplitter under the spacer that is directly below the first PanelSplitter.
    a. Set the Splitter Position and, under the Behavior section of the Property Inspector, set the splitter's Disabled property to True.
    b. Drag the Detail table into the first facet of this PanelSplitter (only the PanelHeader part of the table) and delete the PanelGroupLayout that used to surround it.
15. Insert a Pie Chart into the 2nd facet of the PanelSplitter containing the Detail table by dragging the instance of DPieChart that falls under Dheadersview (the same Dheadersview that had RepairpartsTbl under it).
    a. In the Pie Chart wizard, set the "Pie" to PartsCost and the "Slices" to RecordNsn.
    b. Surround the chart with a PanelHeader and set the text.
        i. Set the Padding Left of the PanelHeader to around 80px.
16. Add a new ShowDetailItem to the PanelTabbed component in the mainContent facet of the page and title it "Unit Totals."
    a. Drag DOUACTotalsVO (under DRACTotalsVO in the Data Control) onto the page as a Master-Detail table.

        i. Set the layout of the PanelGroupLayout that is automatically added with this component to Scroll.
        ii. Change the text of both PanelHeaders.
    b. Surround each PanelHeader (by right-clicking and choosing "Surround With") with a horizontal PanelGroupLayout.
        i. Drag DRACTotals from the Data Control panel into the first PanelGroupLayout (under the existing table's PanelHeader) as a Graph -> Pie Chart.
            1. In the Pie Chart wizard, set the "Pie" to TotalCost and the "Slices" to RegionalActivityCode.
        ii. Drag the instance of DOUACTotals that is underneath DRACTotals in the Data Control into the second horizontal PanelGroupLayout as a Graph -> Pie Chart.
            1. In the Pie Chart wizard, set the "Pie" to UnitTotalCost and the "Slices" to OwnerUnitAddressCode.
            2. Change the height and width using the InlineStyle field in the Property Inspector: "height:300px;width:300px."
        iii. Set SeriesRolloverBehavior in the Property Inspectors of both pie charts to be RB-HIGHLIGHT.
        iv. Set an appropriate Padding Left for both charts.

G. ECHO: GMRT MAINTENANCE PRODUCTION REPORT

Page Layout

The "Main Report" ShowDetailItem for this report is structured with one PanelHeader and one PanelSplitter inside of a main PanelGroupLayout that is set to scroll. The PanelHeader holds the page's Query Panel. Below this, the 1st facet of the vertical PanelSplitter contains a PanelGroupLayout, which surrounds the PanelHeader holding the Master table of the report. The 2nd facet contains a PanelGroupLayout holding a PanelTabbed component.

The PanelTabbed component includes two ShowDetailItems. The first is titled “Tasks & Orders,” and holds a PanelSplitter (inside a PanelGroupLayout). Each facet of this horizontal PanelSplitter holds one Detail table inside a PanelHeader. To the left is a table displaying tasks, while to the right is a table displaying open parts orders.

The second ShowDetailItem, “Order Status Details,” contains a vertical PanelGroupLayout with two PanelHeaders, each separated by a spacer. The first PanelHeader holds a second copy of the orders Detail table, while the second holds a Sub-Detail table.

Figure 77. Top-level structure for the Echo page.

Figure 78. Detailed structure within the “Main Report” tab of the Echo page.

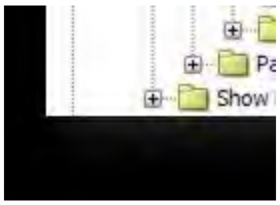


Figure 79. Focusing on the PanelTabbed component.

Basic Steps

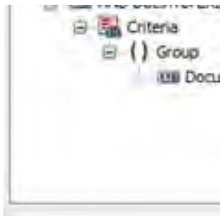
1. Create all view objects.
    a. *Note: DueinTblVO and DueinStatTblVO are simply the default view objects generated by JDeveloper from the DueinTbl and DueinStatTbl entity objects. ETasksVO is a new view object, but can easily be created from SrtasksTblEO using a wizard (must add a WHERE clause).*
    b. Create a read-only view object named ESrTypeListVVO using the Create Business Components from Tables wizard.

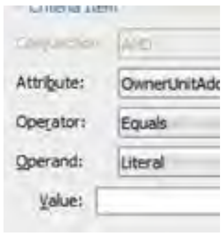
        i. Go to the wizard page that says "Query-Based View Object" and create one based on SrHeadersTbl.
        ii. Exit the wizard and go to the Attributes tab of the view object.
            1. Delete all attributes except ServiceRequestType.
        iii. Go to the Query tab and delete all of the other attributes from the SQL as well.
            1. Add a Group By statement to the query to group based on ServiceRequestType.

2. Create all view links and transient attributes.
    a. Set labels in the UI Hints tab for all four view objects.
    b. In the General tab, adjust the tuning for each of these.
        i. EHeadersDisplayVO:
            1. Range Size = 15
            2. in batches of = 18
        ii. ETasksVO:
            1. Range Size = 20
            2. in batches of = 23
        iii. DueinTblVO:
            1. Range Size = 20
            2. in batches of = 23
        iv. DueinStatTblVO:
            1. Range Size = 10
            2. in batches of = 13
    c. Under the Attribute tab of EHeadersDisplayVO, select the ServiceRequestType attribute.
        i. Go to the LOV tab for this attribute and click to add a new LOV.
            1. Within the wizard, click to add a new view accessor which points to the ServiceRequestType attribute in

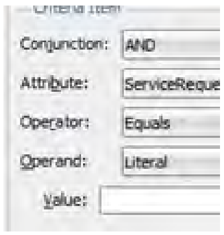
ESrTypeListVVO. The ServiceRequestType field should now show up in forms as a drop-down menu with options.

3. Create a view criteria in EHeadersDisplayVO:

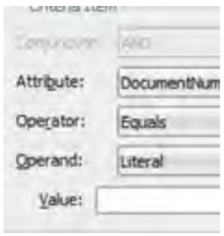

Criteria
( ) Group
Docu


Criteria Item

Conjunction:	AND
Attribute:	OwnerUnitAdc
Operator:	Equals
Operand:	Literal
Value:	


Criteria Item

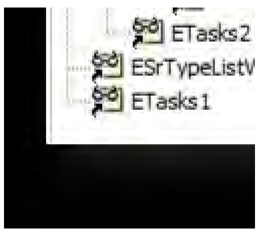
Conjunction:	AND
Attribute:	ServiceReque
Operator:	Equals
Operand:	Literal
Value:	


Criteria Item

Conjunction:	AND
Attribute:	DocumentNum
Operator:	Equals
Operand:	Literal
Value:	

Note: The DocumentNumber attribute must be accessed by first creating a new View Criteria Item based on "DueinTbl." This will generate automatically generate *a group with an Item inside it, and the inner item can then be set to any attribute from DueinTblVO.*

4. Create a new application module for Echo as shown below. Make sure that DueinTbl (with DueinStatTbl beneath it) and Etasks are both placed under the same instance of EHeadersDisplay. This is the instance of EHeadersDisplay that should be used on the page.



5. Create a new page based on the XXMC Reports template and enter the useCaseTitle.
    a. Construct the page's toolbar.
6. Add a PanelTabbed component to the mainContent window and name the ShowDetailItem "Main Report."
7. Add a PanelGroupLayout within the main tab and set it to scroll.
    a. Add a spacer within this.
8. Add a Query Panel to the page by dragging the Named Criteria BHeadersDisplayVOCriteria (under BHeadersDisplay in the Data Control) onto the PanelGroupLayout beneath the spacer.
    a. Adjust the PanelHeader text and Query Panel settings.
9. Add a Master-Detail table to the page under the Query Panel's PanelHeader by dragging the instance of ETasks that is underneath EHeadersDisplay in the Data Control.

    a. Input the ID of the Master table into the ResultComponentID field in the Query Panel's Property Inspector.
    b. Choose the attributes for each table and enable sorting and filtering in the Master table.
        i. Change the Range Size in the Master table's Property Inspector to 20.
    c. Change the names of the PanelHeaders that enclose both tables.
10. Add a second Detail table to the page by dragging the instance of DueinTbl that is under EHeadersDisplay (the same EHeadersDisplay that ETasks is under) and dropping it below the existing tables.
    a. Surround this with a PanelHeader and set its title.
11. Add a vertical PanelSplitter to the page directly under the Query Panel's PanelHeader.
    a. Adjust the Splitter Position.
    b. Drag the PanelHeader holding the Master table into the 1st facet of this PanelSplitter and make sure it is enclosed in a vertical PanelGroupLayout.
    c. In the 2nd splitter facet, insert a vertical PanelGroupLayout.
        i. Add a PanelTabbed component inside this.
12. Title the existing ShowDetailItem of the PanelTabbed component "Tasks & Orders" and put a vertical PanelGroupLayout inside it.
    a. Add a spacer, followed by a horizontal PanelSplitter.
        i. Adjust the Splitter Position.
    b. Drag the PanelHeader containing the tasks (ETasksVO) Detail table into the 1st facet of the horizontal PanelSplitter.
    c. Drag the PanelHeader containing the orders (DueinTblVO) Detail table into the 2nd facet.
13. Add a second ShowDetailItem to the PanelTabbed component and change the text to "Order Status Details."

a. In the “Order Status Details” tab, add a Master-Detail table by dragging the instance of DueinStatus that is underneath DueinTbl (which is also underneath EHeadersDisplay) onto the page.

b. Add a spacer between the PanelHeaders of the two tables and a second spacer above the Master table’s PanelHeader.

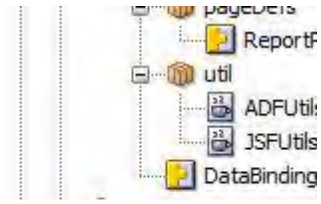
c. Change the text of the two PanelHeaders.

d. Edit the attributes in both tables and enable sorting and filtering.

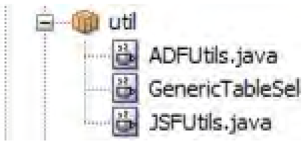
14. Add highlighting functionality, so that when the user clicks a task in the Detail table, the related orders in the second Detail table will automatically be selected.

    a. Copy the code for the 3 Java classes (ADFUtils.java, GenericTableSelectionHandler.java, JSFUtils.java) used in the sample project on this website: http://technology.amis.nl/2012/02/06/adf-11g-fancy-master-detail-or-how-to-highlight-related-detail-records/

        i. Create three new Java classes in JDeveloper (with the same names as above) within the “utils” package under the “Application Sources” folder in the ViewController portion of the application. Paste the appropriate code into each class.



b. Create a new Java class called HighLightBean with directory "beans" so that it will be located under ViewController -> Application Sources -> beans.
   i. Copy the code for this from the same project.
      1. At the beginning of the code, adjust the package appropriately.
      2. Under the method onCountryTableSelect, change the name of the iterator to the iterator of the Tasks table used on the Echo page (you will have to look this up).
      3. Under the method matchEM, change the name of the attribute that will be matched on to "TaskNumber."



c. Open the adfc-config.xml file under ViewController and add a new managed bean with the following settings (and the appropriate directory):
   i. Scope: pageFlow
   ii. Name: HighLightBean
d. Go into the Page Source for Echo and find the part relating to the first Detail table (based on ETasks). Change the following lines:

        i. selectionListener="#{pageFlowScope.HighLightBean.onCountryTableSelect}"
        ii. rowSelection="single"

    e. Select the second Detail table (based on DueinTbl) in the Structure Pane and then go to the Advanced section of the Property Inspector.
        i. Set the Binding field: #{pageFlowScope.HIghLightBean.locTable}
        ii. Set the Row Selection field to "multiple."

H. FOXTROT: DOCUMENT NUMBER REPORT

Page Layout

The "Main Report" ShowDetailItem for this report contains only a vertical PanelSplitter. Within this, the 1st facet holds a PanelGroupLayout (PGL) set to scroll. This PGL contains a spacer followed by a vertical PGL, a horizontal PanelSplitter, and another vertical PGL. In the first vertical PGL is a PanelHeader holding the Query Panel. The second contains a PanelHeader holding the Detail table that displays parts orders. Between these, the PanelSplitter has the Master table in its 1st facet, and a vertical PanelSplitter in its 2nd facet.

The 1st facet of the vertical PanelSplitter holds a PanelHeader with a PanelFormLayout used for displaying details about the Master table. Its 2nd facet contains a horizontal PanelGroupLayout with a spacer, an OutputText component, and a Gauge.

Note: The 2nd facet of the outer (vertical) PanelSplitter was not actually used to hold any content. Therefore, this PanelSplitter does not need to be present; the "Main Report" tab could simply be created with a PanelGroupLayout set to scroll.

The "Orders by NSN" ShowDetailItem on this page contains a PanelGroupLayout with orientation set to scroll. Within this are two vertical

PanelGroupLayouts, the first surrounding the Query Panel's PanelHeader and the second surrounding a results table.

Figure 80. Top-level structure for the Foxtrot page.

Figure 81. Structure of the “Main Report” tab of the Foxtrot page.

Figure 82. Detailed structure of the horizontal PanelSplitter in the “Main Report” tab.

Figure 83. Structure of the “Orders By NSN” tab of the Foxtrot page.

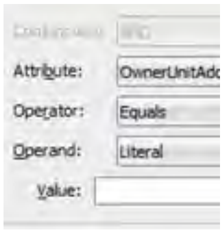
Basic Steps

1. Create a database view, Fheadersview, and add an entity object and a default view object in JDeveloper based off of this view.
2. Create the view object FDueinAll as a Query-based view object in JDeveloper.
3. Create all view links and transient attributes.
    a. Set labels in the UI Hints tab for both of the view objects.
    b. In the General tab, adjust the tuning for each of these.
        i. FheadersviewVO:
            1. Range Size = 20
            2. in batches of = 23
        ii. FDueinAll:
            1. Range Size = 20
            2. in batches of = 23
    c. Click on the RecordNsn attribute in the editor for FDueinAll and set its Width property (under UI Hints) to 20.
        i. Similarly, change the Width of OwnerUnitAddressCode in Fheadersview to 20.
4. Create two view criteria.
    a. In FheadersviewVO:

Group

Doc

AND



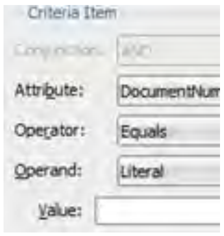
Conjunction: AND

Attribute: OwnerUnitAdd

Operator: Equals

Operand: Literal

Value:



Criteria Item

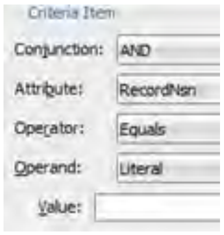
Conjunction: AND

Attribute: DocumentNum

Operator: Equals

Operand: Literal

Value:



Criteria Item

Conjunction: AND

Attribute: RecordNsn

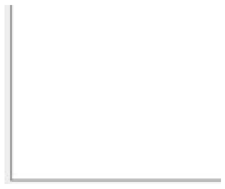
Operator: Equals

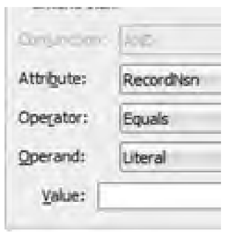
Operand: Literal

Value:

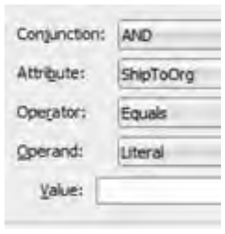
Note: The DocumentNumber attribute must be accessed by first creating a new View Criteria Item based on “FDueinAll.” This will generate automatically generate a group with an Item inside it, and the inner item can then be set to any attribute from FDueinAll. Once this is done, a second item can be added to this same group for the RecordNsn attribute.

b. In FDueinAll:



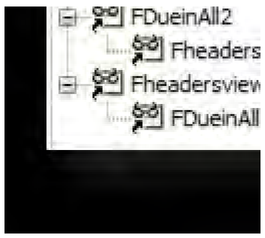


Conjunction: AND
Attribute: RecordNsn
Operator: Equals
Operand: Literal
Value:



Conjunction: AND
Attribute: ShipToOrg
Operator: Equals
Operand: Literal
Value:

5. Create a new application module for Foxtrot as shown here.



6. Create a new page based on the XXMC Reports template and enter the useCaseTitle.
    a. Construct the page's toolbar.

7. Add a PanelTabbed component to the mainContent window and name the ShowDetailItem “Main Report.”
8. Add a PanelGroupLayout within the main tab and set it to scroll *(note: In the existing page, there is a PanelSplitter in the main tab, and the scrollable PanelGroupLayout is inside this. However, because the 2nd facet of this outer PanelSplitter was never used, the component ended up being unnecessary).*
    a. Add a spacer within this.
9. Add a Query Panel below the spacer by dragging and dropping the named criteria FheadersviewVOCriteria under Fheadersview in the Data Control.
    a. Change the PanelHeader text and the Query Panel settings.
    b. Surround the PanelHeader with a vertical PanelGroupLayout (by right-clicking this component in the Structure Pane and choosing “Surround With”) if it was not automatically put in one.
        i. Add a spacer to this PanelGroupLayout beneath the PanelHeader.
10. Add a Master-Detail table to the page under the Query Panel within the outer PanelGroupLayout (the one set to scroll) by dragging the instance of FDueinAll that is underneath Fheadersview.
    a. In both tables, include only the relevant attributes and enable sorting and filtering.
        i. Change the Range Size in the Master table’s Property Inspector to 20.
    b. Change the names of the PanelHeaders that enclose both tables.
    c. Copy the ID of the Master table and paste this into the ResultComponentID attribute in the Property Inspector of the Query Panel.
11. Drag a Panel Splitter into the scroll PanelGroupLayout, directly below the vertical PanelGroupLayout that contains the Query Panel.
    a. Set layout to Horizontal and adjust the Splitter Position.

b. Drag the PanelHeader with the Master table into the 1st facet of the PanelSplitter.

c. Add a second PanelSplitter to the 2nd facet of the horizontal PanelSplitter, and set its orientation to vertical.

    i. Set Splitter Position change Disabled to true in the Property Inspector.

    ii. Adjust the Top Padding and Left Padding properties as needed.

12. Drag Fheadersview from the Data Control panel into the 1st facet of the innermost (vertical) PanelSplitter and create an ADF read-only form.

    a. Surround the PanelFormLayout with a PanelHeader and change its title.

13. Put a Gauge component in the 2nd facet of the innermost (vertical) PanelSplitter by dragging the PercentOfDocumentsClosed column from underneath FHeaderview in the Data Control.

    a. In the menu that appears, choose a horizontal status meter with no titles and no thresholds.

    b. In the wizard, make sure "Percent of Documents Closed" is listed as the metric.

        i. Set the minimum value to 0 and the maximum to 100.

        ii. Leave other settings as they are.

    c. Go to the Property Inspector and in the InlineStyle field, enter "width:150px;height:60px."

14. Surround the Gauge component with a horizontal PanelGroupLayout.

    a. Insert a spacer in the PanelGroupLayout before the Gauge and adjust its Width property.

    b. Add an OutputText component after the spacer and before the gauge and set its Text property to "Status."

15. In the second ShowDetailItem of the PanelTabbed component add a PanelGroup ayout with layout set to scroll.

16. Drag the named view criteria under the FDueinAll into the PanelGroupLayout as a “Query Panel with Table.”
    a. Adjust the text on the Panel Header that surrounds the Query Panel and change the Query Panel settings.
    b. For the results table, change the Range Size to 20.
        i. Adjust the attributes in the results table to match those in the Detail table in the other tab.
    c. Rearrange the Query Panel and table so that each is in its own vertical PanelGroupLayout within the scrollable PanelGroupLayout.
        i. In JDeveloper, view objects based only on a SQL query do not automatically add filter fields to tables, so, if filtering is desired, this must be done manually.
            1. In the Structure Pane, go to the results table and expand any columns that you want to be able to filter on.
                a. Drag an InputText component into the “Filter” facet of each of these and set its value to be #{vs.filterCiteria.*nameOfAttribute*}.
17. Adjust the Query Panel so that it will automatically collapse once the user hits the “Search” button, leaving more room for the table of results.
    a. Set the Disclosed property of the Query Panel using the expression builder #{bindings.Fheadersview1.estimatedRowCount lt 1}.

PAGE LAYOUT

Basic Steps

1. Open the wizard to create a new ADF Page Template.
    a. On the first page of the wizard, enter these settings:
        i. Name: ReportPage

        ii. Type: JSP XML *(must be saved as jspx or you will not be able to use it when creating pages).*

    b. On the second page of the wizard, select “blank template.”

    c. On the remaining pages of the wizard, add the following items:

        i. Facet Definitions

            1. mainContent

            2. Toolbar

        ii. Attributes

            1. useCaseTitle

2. Start the page with a vertical PanelSplitter:

    a. In the Property Inspector, set Splitter Position to 150 (approximately).

        i. Under the Behavior section, set the Disabled property to “True.”

3. Put another vertical PanelSplitter in the bottom facet of the first one:

    a. Set Splitter Position to 28 and Disabled to “True.”

    b. Add a Facet Ref component (referring to Toolbar, which was defined in the Page Template wizard) in the top panel of this PanelSplitter.

4. Insert another PanelSplitter in the top (1st) facet of the outer PanelSplitter with Orientation = Horizontal and Splitter Position = 150 and Disabled = True:

    a. Insert an Image component into the left (1st) facet of this PanelSplitter, which displays the Marine Corps logo: http://learn.shorelineschools.org/shorewood/bbaseball/images/marines.jpg

        i. In the Property Inspector, under the Layout tab of the Style section, set the Height and Width to “auto.”

            1. Under the Border/Outline tab of the Style section, set the Border Color to Navy (#000052) and the Border Radius 80px.
    b. Drag an OutputText component into the right (2nd) facet of this PanelSplitter:
        i. In the Property Inspector, under the Font/Text tab of the Style section, set Color=White, Font Size=800%, Font Family = Arial, and Font Weight = Bold.
            1. Under the Border/Outline tab of the Style section, add a left border (Border Color = Navy) to achieve appropriate spacing from the logo.

5. Add a PanelStretchLayout to the bottom (2nd) facet of the inner vertical PanelSplitter (the one in the bottom facet of the outer PanelSplitter):
    a. Insert Spacers in the Start, End, and Bottom facets of the PanelStretchLayout.
    b. Put a Facet Ref component (referring to mainContent, which was defined in the Page Template wizard) in the center portion of the PanelStretchLayout.
    c. In the Top facet of the PanelStretchLayout, add an OutputText component.
        i. Put in these settings:
            1. Value: #{attrs.useCaseTitle}
            2. Font Family: Arial
            3. Color: Dark navy (#000052)
            4. Font Size: Large
            5. Font Weight: Bold
            6. Add a left border under the Border/Outline tab of the Property Inspector's style section:
                a. Border Left Color: White
                b. Border Left Style: Solid

    c. Border Left Width: 20px

7. Add a top border under the Border/Outline tab of the Property Inspector’s style section:
    a. Border Left Color: White
    b. Border Left Style: Solid
    c. Border Left Width: 10px

I. DASHBOARD (MAIN PAGE)

Page Layout

Basic Steps

1. Create a new JSP XML page titled Dashboard.jspx based on the XXMC Reports template.
2. Click on the af:pageTemplate component in the Structure Pane, and in the Property Inspector enter spaces for the UseCaseTitle (essentially leaving it blank).
3. Do not include anything in the Toolbar facet of the template.
4. Drag a PanelGroupLayout onto the mainContent facet of the template and set its layout to be vertical.
    a. Drag two other PanelGroupLayouts (horizontal) inside this one:
        i. Put three Button components into each of these, with spacers in between:
            1. For each Button, go into the Property Inspector and set the Height to 100px and Width to 180px:
                a. Change the text to be Size = x-large, Font Family = Arial, and Font Weight = bold.
                b. Set Text Color to be navy (#000052).
                c. Set the Text Align property to Center.
                d. Set Line Height to be 80px.
                e. Under the Border/Outline tab of the Style section, set the border on the button to have Border Width = Thick and Border Color = Navy (#000052).
            2. Give each of the buttons a label, Alpha through Foxtrot

Navigation

Page Layout

Basic Steps

1. Open the adfc-config.xml file that is created with the application (under ViewController).
    a. Drag each of the eight pages (two of which are for the Bravo Use Case) associated with the project onto the diagram *(note: There will be nine pages if you create a BravoSelect page, which just displays two buttons to let the user to choose whether he wishes to see service requests with null labor hour fields or not in Bravo. This will direct them to either the normal BravoQuery page or the Bravo2 no-null page).*

    i. For each of the page icons on your diagram, drag a Wildcard Control Flow Rule from the Component Palette onto the diagram *(note: If the user has a BravoSelect page, he only needs one Wildcard Control Flow Rule for all three pages related to Bravo—it should always direct the user to this gateway page).*
    ii. Drag a Control Flow Case arrow onto the diagram to connect each Wildcard Control Flow Rule to a page *(see previous diagram; the Wildcard Control Flow Rule for Bravo should connect to the BravoSelect page).*
        1. Name each of these arrows with the name of the Use Case page it is pointing towards (in lowercase).
    iii. Drag a Control Flow Case arrow onto the page so that it points from BravoSelect to BravoQuery (the page with nulls) and give it a name.
    iv. Drag a second Control Flow Case arrow onto the page so that it points from BravoSelect to Bravo2 (the page that filters out service requests with null data) and give it a name, like "nonull."

2. Open each report page that is part of the application and click on each of the navigation buttons on the page in turn (these will be in the Toolbar facet at the top of the page).
    a. Go to the Property Inspector for the selected button and set the Action property. There should be a drop-down menu that will list the names of all of the Use Case pages in lowercase; choose the one that corresponds to the text on the button, which indicates which page that the button should navigate to *(note: Any buttons labeled "Bravo" should direct to the BravoSelect page by selecting Action = bravo)*. Each page should also have a button labeled "Dashboard," and its Action property should be set to "dashboard."

3. Do the same for each of the six buttons on the main Dashboard page.
4. Open the BravoSelect page, and set the Action property for the two buttons on the page so that one directs to BravoQuery (Action = blank) and the other directs to Bravo2 (Action = nonull), as appropriate.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Acquisition Category (ACAT). (n.d.). *ACQupedia*.[Online], Available: https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=a896cb8a-92ad-41f1-b85a-dd1cb4abdc82. Accessed Nov. 21, 2014.

2. Global Combat Service Support-Marine Corps. (n.d.). USMC. [Online]. Available: http://www.marcorsyscom.usmc.mil/sites/gcss-mc/index.aspx/benefits. Accessed Nov. 21, 2014.

3. *Marine Corps Coordinated Secondary Reparable Management (CSM) Program,* MCO 4400.200, Commandant of the Marine Corps (I&L), Washington, DC, 2012, pp. 1–7.

4. *Ground Equipment Maintenance Program (GEMP)*, MCO 4790.25, Commandant of the Marine Corps (I&L), Washington, DC, 2014, pp. 4–24.

5. *The National Stock Number (NSN), The Gear that Keeps the Supply Chain Running,* Defense Logistics Agency (DLA), Battle Creek, MI, 2010, pp. 1–12.

6. Oracle's History Innovation Leadership Results. (2014, June 9). Oracle Corporation. [Online]. Available: http://www.oracle.com/us/corporate/history/index.html

7. M. Blackmore., et al. (2010, June). *Building Agile Applications Using Fusion Development and Oracle Enterprise Architecture Principles.* Oracle Corporation. Redwood City, CA. [Online]. Available: http://www.oracle.com/technetwork/topics/fusion-development-oea-091563.html

8. *Oracle Fusion Applications Administrator's Troubleshooting Guide, 11g Release 1 (11.1. 4),* Oracle Corporation, Redwood City, CA, [Online]. Available: https://docs.oracle.com/cd/E28271_01/doc.1111/e25450/toc.htm

9. L Jamen. (2011, Jan.). *Oracle Fusion Middleware: Concepts Guide 11g Release (11.1.1),* Oracle Corporation, Redwood City, CA. [Online]. Available: http://docs.oracle.com/cd/E17904_01/core.1111/e10103.pdf

10. R. Eckstein. (2007, Mar.). Java SE Application Design With MVC. [Online]. Available: http://www.oracle.com/technetwork/articles/javase/index-142890.html#2

11. L. Akel. (2007, April). *A. D. F. 11g Primer, Introduction to the building blocks of a Fusion Web application: An Oracle White Paper.* Oracle Corporation. Redwood City, CA. [Online]. Available: http://www.oracle.com/technetwork/testcontent/oracle-adf-11g-primer-154277.pdf

12. S. O'Brien and S. Shmeltzer. (2011, June). *Oracle Application Development Framework Overview: A*n Oracle White Paper. Oracle Corporation. Redwood City, CA. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/adf/adf-11-overview-1-129504.pdf

13. *Oracle ADF Data Sheet.* Oracle Corporation, Redwood City, CA. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/adf/adf11g-data-sheet-1-133847.pdf

14. R. Gordon., et al. (2011, Aug.). *Oracle Fusion Middleware—Fusion Developer's Guide for Oracle Application Development Framework 11g Release 1 (11.1. 1.5.0),* Oracle Corporation, Redwood City, CA. [Online]. Available: https://docs.oracle.com/cd/E25054_01/web.1111/b31974.pdf

15. G. Ronald, *Quick Start Guide to Oracle Fusion Development: Oracle JDeveloper and Oracle ADF.* New York, NY: McGraw-Hill, 2010, pp.31–32.

16. *Oracle JDeveloper Data Sheet.* Oracle Corporation, Redwood City, CA. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/jdev/jdeveloper11g-datasheet-1-133040.pdf

17. J. Purushothaman, *Oracle ADF Real World Developer's Guide*. Birmingham, UK: Packt Publishing Ltd., 2012, pp. 19–20.

18. D. Mills, P. Koletzke, and A. Roy-Faderman. *Oracle JDeveloper 11g Handbook*. McGraw-Hill, Inc., New York City, 2010, pp. 62–65.

19. *Oracle Fusion Middleware: Understanding Oracle WebLogic Server 12c (12.1.3).* Oracle Corporation. Redwood City, CA. [Online]. Available: http://docs.oracle.com/cd/E24329_01/web.1211/e24446/toc.htm

20. M. Schildmeijer. (2001). *Oracle WebLogic Server 11gR1 PS2: Administration essentials*. Packt Publishing Ltd., Mumbai, India, pp. 12–13, 58–59, 162–164.

21. S. Harper. (2008, Sept.). Oracle SQL Developer for Database Developers: An Oracle White Paper. Oracle Corporation. Redwood City, CA. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/sql-developer/sqldeveloperwhitepaper-v151-130908.pdf